日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００２年１０月　２日

出願番号
Application Number: 特願２００２－２９００５０

[ST.10/C]: [ＪＰ２００２－２９００５０]

出願人
Applicant(s): インターナショナル・ビジネス・マシーンズ・コーポレーション

２００３年　２月　７日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３００５３６３

【書類名】　特許願

【提出日】　平成14年10月 2日

【整理番号】　JP9020151

【あて先】　特許庁長官　殿

【国際特許分類】　G06F 19/00

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

【氏名】　戸澤　晶彦

【発明者】

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　東京基礎研究所内

【氏名】　村田　真

【特許出願人】

【識別番号】　390009531

【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】　100086243

【弁理士】

【氏名又は名称】　坂口　博

【代理人】

【識別番号】　100091568

【弁理士】

【氏名又は名称】　市位　嘉宏

【代理人】

【識別番号】　100108501

【弁理士】

【氏名又は名称】　上野　剛史

【復代理人】

【識別番号】　100110607

【弁理士】

【氏名又は名称】　間山　進也

【手数料の表示】

【予納台帳番号】　062651

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9706050

【包括委任状番号】　9704733

【包括委任状番号】　0207860

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　文書検索システム、文書検索方法、文書検索を実行するためのプログラム、および該プログラムが記録されたコンピュータ可読な記憶媒体、コンパイル装置、コンパイル方法、コンパイル方法を実行するためのプログラムおよび該プログラムが記録されたコンピュータ可読な記憶媒体、検索オートマトン評価装置

【特許請求の範囲】

【請求項１】　要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索システムであって、前記文書検索システムは、

入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードを識別して検索オートマトンを生成するコンパイル装置と、

前記コンパイル装置により生成された検索オートマトンを記憶する記憶装置と、

前記記憶装置から前記検索オートマトンを読み出して記憶すると共に前記文書を読み込み、前記文書に含まれる前記要素識別子の種類の異なる複数のノードの状態と前記検索オートマトンとを使用してストリーム検索を実行し、検索されたノードを出力する検索オートマトン評価装置とを含む文書検索システム。

【請求項２】　前記検索オートマトン評価装置は、識別された要素識別子に対応して左ノードおよび下ノードを記憶させ、前記左ノードおよび前記下ノードの検索結果を使用して前記検索オートマトンを評価することにより、その時点で判断しているノードの状態遷移を判断する、請求項１に記載の文書検索システム。

【請求項３】　前記コンパイル装置は、初期状態と終了状態と検索状態とに対応した状態遷移が登録された検索オートマトンを生成する、請求項１に記載の文書検索システム。

【請求項４】　要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索方法であって、前記文書検索方法は、

コンパイル装置により入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードを識別して検索オートマトンを生成するステップと、

前記コンパイル装置により生成された検索オートマトンを記憶装置に記憶させるステップと、

前記記憶装置から前記検索オートマトンを読み出して記憶すると共に前記文書を読み込み、前記文書に含まれる前記要素識別子の種類の異なる複数のノードの状態と前記検索オートマトンとを使用して検索オートマトン評価装置を使用してストリーム検索を実行するステップと

を含む文書検索方法。

【請求項５】　前記ストリーム検索を実行するステップは、識別された要素識別子に対応して左ノードおよび下ノードを記憶させ、前記左ノードおよび前記下ノードの検索結果を使用して前記検索オートマトンを評価することにより、その時点で判断しているノードの状態遷移を判断するステップを含む、請求項４に記載の文書検索方法。

【請求項６】　前記検索オートマトンを生成するステップは、初期状態と終了状態と検索状態とに対応した状態遷移が登録された検索オートマトンを生成するステップを含む、請求項４に記載の文書検索方法。

【請求項７】　要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索方法を実行するためのコンピュータ実行可能なプログラムであって、前記プログラムは、コンピュータに対して、

入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードを識別して検索オートマトンを生成するコンパイル装置として機能させるステップと、

前記コンパイル装置により生成された検索オートマトンを記憶装置に記憶させるステップと、

前記記憶装置から前記検索オートマトンを読み出して記憶すると共に前記文書を読み込み、前記文書に含まれる前記要素識別子の種類の異なる複数のノードの状態と前記検索オートマトンとを使用してストリーム検索を実行する検索オートマトン評価装置として機能させるステップと

を実行させるプログラム。

【請求項８】　前記ストリーム検索の実行は、識別された要素識別子に対応し

て左ノードおよび下ノードを記憶させ、前記左ノードおよび前記下ノードの検索結果を使用して前記検索オートマトンを評価することにより、その時点で判断しているノードの状態遷移を判断する、請求項７に記載のプログラム。

【請求項９】　前記検索オートマトンを、初期状態と終了状態と検索状態とに対応した状態遷移が登録された検索オートマトンとして生成する、請求項７に記載のプログラム。

【請求項１０】　要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索方法を実行するためのコンピュータ実行可能なプログラムが記録されたコンピュータ可読な記憶媒体であって、前記プログラムは、コンピュータに対して、

入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードを識別して検索オートマトンを生成するコンパイル装置として機能させるステップと、

前記コンパイル装置により生成された検索オートマトンを記憶装置に記憶させるステップと、

前記記憶装置から前記検索オートマトンを読み出して記憶すると共に前記文書を読み込み、前記文書に含まれる前記要素識別子の種類の異なる複数のノードの状態と前記検索オートマトンとを使用してストリーム検索を実行する検索オートマトン評価装置として機能させるステップと

を実行させる記憶媒体。

【請求項１１】　前記ストリーム検索の実行は、識別された要素識別子に対応して左ノードおよび下ノードを記憶させ、前記左ノードおよび前記下ノードの検索結果を使用して前記検索オートマトンを評価することにより、その時点で判断しているノードの状態遷移を判断し、前記検索オートマトンを、初期状態と終了状態と検索状態とに対応した状態遷移が登録された検索オートマトンとして生成する、請求項１０に記載の記憶媒体。

【請求項１２】　文書検索を行うための検索オートマトンを生成するためのコンパイル装置であって、前記コンパイル装置は、入力された検索式を検索の意味の等価性を保存しつつ逆方向の軸を含む軸および連言や否定を含む論理式を置換

することにより状態遷移を生成および登録し、前記後方ノードの複数の状態と遷移条件と少なくとも検索状態とを含む検索オートマトンを生成する、コンパイル装置。

【請求項１３】　前記コンパイル装置は、前記後方ノードを前記要素識別子の種類に関連して左ノードおよび下ノードとして識別し、前記複数の状態は、前記左ノードおよび前記下ノードの状態である、請求項１１に記載のコンパイル装置。

【請求項１４】　文書検索を行うための検索オートマトンを生成するためのコンパイル方法であって、前記コンパイル方法は、

入力された検索式を検索の意味の等価性を保存しつつ逆方向の軸を含む軸および連言や否定を含む論理式を置換することにより状態遷移を生成および登録し、前記後方ノードに対応させて前記後方ノードの複数の状態を記憶装置に記憶させるステップと、

前記後方ノードの複数の状態と遷移条件と少なくとも検索状態と、到達状態とを対応させて前記記憶装置に登録して検索オートマトンを生成するステップと

を含む、コンパイル方法。

【請求項１５】　前記コンパイル方法は、前記後方ノードを前記要素識別子の種類に関連して左ノードおよび下ノードとして識別するステップを含み、前記複数の状態は、前記左ノードおよび前記下ノードの状態である、請求項１３に記載のコンパイル方法。

【請求項１６】　文書検索を行うための検索オートマトンを生成するためのコンパイル方法をコンピュータに実行させるためのプログラムであって、前記プログラムは、コンピュータに対して、

入力された検索式を検索の意味の等価性を保存しつつ逆方向の軸を含む軸および連言や否定を含む論理式を置換することにより状態遷移を生成および登録し、前記後方ノードに対応させて前記後方ノードの複数の状態を記憶装置に記憶させるステップと、

前記後方ノードの複数の状態と遷移条件と少なくとも検索状態と、到達状態のとを対応させて前記記憶装置に登録して検索オートマトンを生成するステップと

を実行させる、プログラム。

【請求項１７】　前記プログラムは、前記後方ノードを前記要素識別子の種類に関連して左ノードおよび下ノードとしてコンピュータに識別させるステップを含み、前記複数の状態は、前記左ノードおよび前記下ノードの状態である、請求項１５に記載のプログラム。

【請求項１８】　文書検索を行うための検索オートマトンを生成するためのコンパイル方法をコンピュータに実行させるためのプログラムが記録されたコンピュータ可読な記憶媒体であって、前記プログラムは、コンピュータに対して、

入力された検索式を検索の意味の等価性を保存しつつ逆方向の軸を含む軸および連言や否定を含む論理式を置換することにより状態遷移を生成および登録し、前記後方ノードに対応させて前記後方ノードの複数の状態を記憶装置に記憶させるステップと、

前記後方ノードの複数の状態と遷移条件と少なくとも検索状態と、到達状態のとを対応させて前記記憶装置に登録して検索オートマトンを生成するステップと

を実行させる、記憶媒体。

【請求項１９】　前記プログラムは、前記後方ノードを前記要素識別子の種類に関連して左ノードおよび下ノードとしてコンピュータに識別させるステップを含み、前記複数の状態は、前記左ノードおよび前記下ノードの状態である、請求項１７に記載の記憶媒体。

【請求項２０】　要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索システムであって、前記文書検索システムは、

入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードに対して割り当てられる少なくとも２状態を読み込んで状態遷移を可能とする２状態入力オートマトンを生成するコンパイル装置と、

前記２状態入力オートマトンを記憶する記憶装置と、

前記記憶装置から２状態入力オートマトンを読み出して記憶すると共に前記文書を読み込み、前記２状態を識別して状態遷移を可能とするオートマトン評価装置とを含む文書検索システム。

【請求項２１】　前記２状態は、識別された要素識別子に対応して生成された

木構造の左ノードおよび下ノードの状態であり、前記２状態入力オートマトンは前記オートマトン評価装置の３つの状態を使用する、請求項１９に記載の文書検索システム。

【請求項２２】　要素識別子により要素が区切られた階層構造を有する文書を検索するための検索オートマトンを評価する検索オートマトン評価装置であって、該検索オートマトン評価装置は、

コンパイル装置により生成された複数の入力を同時に判断可能とする検索オートマトンを記憶装置から読み込んで記憶する手段と、

前記文書に含まれる前記要素識別子の種類の異なる複数の入力を識別する手段と、

前記識別された入力と前記検索オートマトンに登録された複数の入力とを使用して検索状態を含む３つの状態間で状態遷移を割り当てるための手段と

を含む、検索オートマトン評価装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、概ね文書検索に関し、より詳細には、式で指定された検索する文書の要素（ノード）を検索オートマトンを使用してストリームベースで検索を実行させる、文書検索システム、文書検索方法、該文書検索方法をコンピュータに対して実行させるためのプログラム、および該文書検索方法をコンピュータに対して実行させるプログラムが記憶されたコンピュータ可読な記憶媒体に関する。また、本発明は、上述した文書検索システムにおいて検索オートマトンを生成するためのコンパイル装置、コンパイル方法、該コンパイル方法をコンピュータに対して実行させるためのプログラム、および該プログラムを記憶したコンピュータ可読な記憶媒体に関する。さらに本発明は、上述した文書検索に使用される検索オートマトン評価装置に関する。

【０００２】

【従来の技術】

コンピュータなどにおいて処理される文書としては、テキスト形式、ハイパー

テキスト・マークアップ・ランゲージ（Hypertext Markup Language:HTML）、SGML（Standard Generalized Markup Language）、次世代HTMLとして注目されているXML（Extensible Markup Language）などが知られている。これらの文書のうち、HTML、SGML、XMLといった文書は、タグとして参照される要素識別子＜や、／＞を使用した階層構造を含み、テキスト形式よりも多くの情報を含ませることが可能となっているため、コンピュータにおいてますます多用されてきている。これらの階層構造を含む文書を効率的に検索するため、一般的には検索式を使用し、その検索式に該当する要素を含む文書およびノードを検索する方法が知られている。例えば検索式としては、特にXML文書を対象とするXpath式が知られている。

【０００３】

XPath式とは、"/"(スラッシュ,slash)で区切られた、要素、属性に対する条件(正確には位置ステップ；location step)の列からなる文字列を含んで構成される。たとえば/html/body/pは、html、body、pという３つの条件からなる XPath式である。この場合、html、body、pは、それぞれ要素の名前(タグ名，tag name）に関する条件となる。このXPath式 /html/body/pをあるHTML文書に対して評価すると、html要素の直下のbody要素の直下に存在するp要素が検索される。一般には、このようなp要素は、木の中に複数存在する。一般にひとつのXPath式は、XML文書の複数のノードの集合を検索する。

【０００４】

XPath式には、タグ名に関する条件のほかに軸(axis)を指定することができる。例えば、もし/htmlのように軸が何も指定されていない場合、軸はchild、すなわちノードの木構造において直下の(子供の)要素を指し示すことになる。XPath式において軸を指定する場合、/decendant::p のような構文によって指定することができる。このdescendantは、子孫要素を示す。軸を/descendant::pとして定める場合には、直下の要素に限らず、木構造の子孫にあたるすべてのpが検索できる。この軸descendantに関しては、より簡易に//pのように、省略形で記述することができる。

【０００５】

XPath式にはタグ名に関する条件、軸に加えてさらに、述語(predicate)を指定することができる。述語は、XML木のノードを検索する際に、そのノードにおいて満たされるべき条件を記述する。述語は、and、or、not などによって論理的に連結することができる。

【０００６】

上述したXPathの仕様は、W3C仕様がこれまで提案されている。W3C仕様のXPathに忠実なXPath評価システムとしては、これまでXalanなどが知られており、この評価システムは、XPathの軸、述語などをすべて評価することができる。これらのXpath評価システムは、DOM、またはそれに類似するデータ構造を用い、XML文書をすべてメモリに展開することによりコンピュータなどにおいて実装されている。DOMは、XML文書の木構造をメモリの中にすべて展開するXML操作インタフェースとして知られている。またこれとは別にSAXとして参照されるシステムおよび方法も知られている。SAXは、XML文書を文書の先頭から順にイベントの形で読出すインタフェース(API)である。以後、本発明においては、SAXのように文書の先頭から順にXML文書を読み出すアプリケーション・インタフェイス（API）を総称して、ストリームベースAPIとして参照する。

【０００７】

上述したストリームベースAPIを使用して文書の先頭から順に読み出しを実行させる文書検索システムについて考察すると、XMLを木構造として表現すれば多くの場合、（１）深さの優先順、（２）ノードの並びについては、左から右の順で読出しが実行される。SAXベースのXPath評価システムとしては、http://XML.coverpages.org/ni2002-08-26-d.htmlに開示されたXMLTKなどが提案されている。しかしながら、従来知られたストリームベースの評価システムは、Xpathにおいて許容されている論理仕様を完全に満たす範囲を処理することができないといった不都合がある。上述した不都合を改善するための試みは種々行われており、例えばD. Olteanuらは、XPath: Looking Forward, http://www.cis.uni-muenchen.de/people/
Meuss/Pub/XMLDM02.pdfにおいて、逆方向の軸をXPathから取り除くことを検討しているものの、XPathの仕様を完全に満たす範囲の論理を処理する点では不充分

であった。

【０００８】

一方で、記号を保持した記憶手段と、記憶手段に記述された記号を読み取る装置と状態制御装置とから構成されるオートマトンという技術的概念も知られている。オートマトンは、記憶手段に書き込まれた記号を読み込み、状態制御装置の前の状態と読み込んだ記号とによって、内部の状態を遷移させ、状態制御装置が保持する最終状態と遷移状態とが一致した場合に処理を終了させるものとされている。オートマトンのコンピュータにおける実装を考えた場合、オートマトンは、いくつかの状態遷移からなるテーブル構造を含んで実装されることになる。上述したオートマトンの基礎については、例えば代表的な文献、Ｊ．ホップクロフト・Ｊ．ウルマン（野崎・高橋・町田・山崎訳）「オートマトン 言語理論 計算論Ｉ、ＩＩ」，サイエンス社,1986年に詳細な説明がなされている。

【０００９】

また、検索オートマトンの用語は、本発明においてオートマトンのある状態を検索状態として他の状態から識別させ、他の状態とは異なる状態制御を実行させるオートマトンのことを意味する。このような検索オートマトンについては、これまでNeven (F. Neven, Design and Analysis of Query Languages for Structured Documents, PhD Thesis, Limburgs Universitair Centrum, 1999) などにより提案されている。しかしながらNevenの方法は、検索オートマトンの性質などを調べたものであり、検索オートマトンのストリームベースでの文書検索システムへの適用性およびその具体的構成については何ら示唆するものではない。

【００１０】

上述した従来の評価装置において、W3C仕様に忠実なXalanなどの評価装置は、主にDOMを用いて構成されており、この理由としては下記の理由を挙げることができる。

１．XPath式//a/ancestor::bは、蓄積された文書のどこかにあるa要素のさらにその祖先のb要素を指し示すXPath式である（上記式中、//aは、/descendant::aの略記である。）。このようなXPath式は、一般に木構造を下りてから上る動作で評価されるため、先頭から順にXML文書を読み出す方式では評価できない。

２．XPath式//a/preceeding-sibling::bは、蓄積されたXML文書のどこかにあるa要素のその弟要素のb要素を指し示すXPath式である。このXPath式では、木構造を右に動いてから左に戻る動作で評価を実行するため、文書の先頭から順にXML文書を読み出す方式では評価できない。

３．XPath式//a[.//b]は、a要素であって、その子孫にb要素が存在するXML文書をすべて選択する。このようなXPath式は、（ｉ）a要素を探した後、その子孫の中で（２）b要素の存在を調べに行く、という手順で評価される。この評価は、順序としては文書の先頭から順に調べてゆく順で可能である。しかしながら、結果として選択されるのはa要素であるため、a要素がどこに存在したかという情報を記憶しておかなくてはならない。

４．XPath式//a[.//b and .//c]は、a要素であって、その子孫にb要素およびa要素が存在するものを選択する。このようなXPath式は、（ｉ）a要素を探した後、その子孫の中で（ｉｉ）b要素の存在を調べにゆき、 最後に（ｉｉｉ）c要素の存在を調べにゆくという手順で評価が行われる。この評価も文書の先頭から順にXML文書を調べるという順では実行できない。

【００１１】

上記理由４の条件は、連言的(conjunctive)な条件として参照される。一方、XPath式 //a[.//b or .//c]は、a要素であってその子孫にb要素またはc要素が存在するものを選択する。このXPath式は、選言的(disjunctive)な条件として参照される。選言しか含まないXPath式は、即座に.//a[.//*[name() = "b" or name() = "c"]] のように書き換えることができる。このため、（ｉ）a要素を探した後、その子孫の中で（ｉｉ）名前がbまたはcであるノードにであうまで検索をする、という手法で評価が実行できる。

【００１２】

従って従来のストリームベースの文書検索は、式が特殊な軸(理由１および２として参照した)や、特殊な述語(理由３および４と関連する)を含まない場合には、XPath式を文書の先頭から順に評価することは容易である。

【００１３】

ところで上述したように従来のDOMを使用する文書評価システムが知られてい

るものの、DOMは、与えられたXML文書をすべてメモリ上に展開することを必要とするので、ストリームベースのSAXに比較してメモリ効率が低く、また実行効率が悪いという不都合がある。また、DOMを使用した評価システムについて、ノードを見ようとしたときにはじめて文書の必要な部分を読み込む処理を実行させる手法を利用する評価システムも想定できる。しかしながら、このような評価装置でもDOMは、いったん読み込んで構築したXML文書の木構造を捨て去ることはしない。XML文書の中には、数ギガバイトのメモリに入りきらないようなものも存在し、このようなXML文書に対してDOM木をメモリ上に作成して保存させておくことは、ハードウエア・リソースの点から現実的なものではなく、文書検索システムの適用性を制限する要因となっていた。

【００１４】

さらに、DOM木を用いるようなナイーブなアルゴリズムでは、上述のように、文書の同じ場所を何回も繰り返し調べることになる。これは、XPathの評価アルゴリズムとしては効率が悪く、たとえば//a[.//b]の評価において、あるa要素の子孫ノードにbがあるかどうかを調べると同時に、そのノードが検索されるべきもうひとつのa要素であるかを調べることができれば文書検索の効率化を達成することが可能となる。

【００１５】

一方で、XPath式が特殊な軸や、述語を持たない場合には、XPath式を、SAXなどのイベント駆動処理系による評価が容易に実行できることは上述した通りである。しかしながら、上述した理由１～４に挙げたように、従来のストリーム検索の技術の手法は、軸や述語の表現のすべてを解釈できず、（ア）述語の中に子供に関する条件しか書けない点(例: XPath 式 //a[.//b]が記述できない)、（イ）XMLTKでは、現在のところ述語の評価を避けている点、（ハ）SGMLに対する有名な文書変換装置としてhttp://www.tas.co.jp/XML/tools/omni/omnimarkに開示されたomnimarkが知られているものの、omnimarkであっても、２つの条件のandを記述してパス評価を行うことができない点(例: //a[.//b and .//c] は記述できない。)などの不都合がある。

【００１６】

また、上述した不都合を改善するために、Olteanuらによるようにxpath式から逆方向の軸を取り除く手法を使用し、例えばXPath式である//a[ancestor::b]からancestor軸を取り除くことにより等価なXPath式として、//b//aを得ることもできる。しかしながら上述した方法は汎用的な方法ではなく、例えばXPath式 //a[not(ancestor::b)]から逆方向の軸ancestor軸を取り除くと、まったく別の結果を与えることになってしまう。上述した点で、現状ではこれらをすべてサポートしたW3Cの仕様に忠実な評価装置をストリームベースAPIによって評価する文書検索システムおよび文書検索方法は知られていない。

【００１７】

さらに、DOMを用いる検索システムでもSAXを使用する検索システムでも、二つの条件をandにより結合させてパス評価を行うことができないといった不都合があった。また、これまでの検索システムでは、兄弟要素に対する軸(following-sibling, preceeding-sibling, etc.)含めたパス評価を行うことができないといった不都合があった。加えてこれまでの評価手法は、述語を満たすノードが存在しないことを示す否定“not”を取り扱うことができないといった不都合があった。

【００１８】

【発明が解決しようとする課題】

本発明は、上記従来技術が含む不都合に鑑みてなされたものであり、本発明は、これまで軸や述語の表現をすべて解釈でき入力される検索式に対して高い汎用性を有し、かつ高い効率で評価を実行することを可能とし、さらには省ハードウエア資源を達成することが可能なストリームベースAPIに基づいた文書検索を可能とすることを目的とする。以下、本発明においてはストリームベース検索を単に、ストリーム検索として参照する。

【００１９】

【課題を解決するための手段】

本発明は、文書検索システムをオートマトンを用いて構成し、検索を文書検索システムの状態遷移のうち、要素識別子の種類を判断して複数の異なる木構造を特定し、複数の異なる木構造の複数の結果を入力とし、初期状態、終了状態の他

、検索状態の間の３つの状態間で状態遷移を可能とする検索オートマトンを構成すれば、従来の文書検索システムの不都合を改善することができる、という新規な着想の下になされたものである。本発明の文書検索システムは、検索状態を識別し、検索状態において初期状態および終了状態とは異なる処理を実行させる。検索オートマトンは、入力された検索式をコンパイルして、初期状態、終了状態および検索状態を状態集合を含んで構成されるテーブル構造によって表現される。本発明においては、要素識別子の種類を判断して前方ノードおよび後方ノードを定義する。さらに後方ノードを複数の木構造に識別し、状態遷移を木構造の複数の後方ノードの状態を入力することにより生じさせるものとする。

【００２０】

本発明では、検索オートマトンは、入力のストリームを解釈し、各ノードに対してノードを解釈しながらオートマトンを実行し、ノードを解釈した際のオートマトンの内部状態を記憶する。この記憶を、本発明においては、ノードに対する状態割当てとして参照する。上述した記述の下では、所定のノードに対する状態割り当てが検索状態の場合に、ノードが検索されたものとされる。本発明で使用する検索オートマトンは、特殊な軸や述語をもった検索式でも判断し、処理を実行することができる。

【００２１】

本発明の文書検索システムが含む検索オートマトン評価装置は、木のいくつかの後方ノードから得た状態の組に対して、遷移後の状態を計算するという動作を実行する。検索オートマトン評価装置は、遷移と同時に、検索される可能性のあるノードをメモリに保持させる。検索オートマトンは、所与のノードに検索状態とされることが確定すれば、そのノードをメモリから検索出力として出力させ、出力されたノードを削除することができる。

【００２２】

一方、検索オートマトン評価装置は、検索されないことが確定した時点でそのノードの情報をメモリから削除しても、すでに検索状態に遷移したノードは記憶されているので文書検索には不都合を生じない。これは、本発明においては終了状態として検索結果を出力させるのではなく、検索状態を含ませた検索オートマ

トンを構成させることにより、検索結果を与えることを可能とする機能が与えられるためである。本発明の文書検索システムを採用することによって、メモリ効率が改善されると同時に、木の同じ部分を一回しかたどる必要しかなく、同時に任意の数の論理的に結合された検索式を評価することが可能となる。

【００２３】

本発明の上述した機能は、コンピュータに対して検索オートマトンを生成させるコンパイル装置により実現され、本発明における検索オートマトンへのコンパイル装置は、特殊な軸を判断し、読み取られた述語を検索オートマトンの各状態遷移に置換することにより、検索オートマトンを生成する。

【００２４】

すなわち、本発明によれば、要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索システムであって、前記文書検索システムは、

入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードを識別して検索オートマトンを生成するコンパイル装置と、

前記コンパイル装置により生成された検索オートマトンを記憶する記憶装置と、

前記記憶装置から前記検索オートマトンを読み出して記憶すると共に前記文書を読み込み、前記文書に含まれる前記要素識別子の種類の異なる複数のノードの状態と前記検索オートマトンとを使用してストリーム検索を実行し、検索されたノードを出力する検索オートマトン評価装置とを含む文書検索システムが提供される。

【００２５】

また、本発明においては、前記検索オートマトン評価装置は、識別された要素識別子に対応して左ノードおよび下ノードを記憶させ、前記左ノードおよび前記下ノードの検索結果を使用して前記検索オートマトンを評価することにより、その時点で判断しているノードの状態遷移を判断することが好ましい。

【００２６】

さらに本発明における前記コンパイル装置は、初期状態と終了状態と検索状態とに対応した状態遷移が登録された検索オートマトンを生成する、請求項１に記

載の文書検索システム。

【００２７】

さらに、本発明によれば、要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索方法であって、前記文書検索方法は、

コンパイル装置により入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードを識別して検索オートマトンを生成するステップと、

前記コンパイル装置により生成された検索オートマトンを記憶装置に記憶させるステップと、

前記記憶装置から前記検索オートマトンを読み出して記憶すると共に前記文書を読み込み、前記文書に含まれる前記要素識別子の種類の異なる複数のノードの状態と前記検索オートマトンとを使用して検索オートマトン評価装置を使用してストリーム検索を実行するステップと

を含む文書検索方法が提供される。

【００２８】

また、本発明によれば、要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索方法を実行するためのコンピュータ実行可能なプログラムであって、前記プログラムは、コンピュータに対して、

入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードを識別して検索オートマトンを生成するコンパイル装置として機能させるステップと、

前記コンパイル装置により生成された検索オートマトンを記憶装置に記憶させるステップと、

前記記憶装置から前記検索オートマトンを読み出して記憶すると共に前記文書を読み込み、前記文書に含まれる前記要素識別子の種類の異なる複数のノードの状態と前記検索オートマトンとを使用してストリーム検索を実行する検索オートマトン評価装置として機能させるステップと

を実行させるプログラムが提供される。

【００２９】

さらに、本発明によれば、要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索方法を実行するためのコンピュータ実行可能なプログラムが記録されたコンピュータ可読な記憶媒体であって、前記プログラムは、コンピュータに対して、

入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードを識別して検索オートマトンを生成するコンパイル装置として機能させるステップと、

前記コンパイル装置により生成された検索オートマトンを記憶装置に記憶させるステップと、

前記記憶装置から前記検索オートマトンを読み出して記憶すると共に前記文書を読み込み、前記文書に含まれる前記要素識別子の種類の異なる複数のノードの状態と前記検索オートマトンとを使用してストリーム検索を実行する検索オートマトン評価装置として機能させるステップと

を実行させる記憶媒体が提供される。

【００３０】

また、本発明によれば、文書検索を行うための検索オートマトンを生成するためのコンパイル装置であって、前記コンパイル装置は、入力された検索式を検索の意味の等価性を保存しつつ逆方向の軸を含む軸および連言や否定を含む論理式を置換することにより状態遷移を生成および登録し、前記後方ノードの複数の状態と遷移条件と少なくとも検索状態とを含む検索オートマトンを生成する、コンパイル装置が提供される。

【００３１】

本発明における前記コンパイル装置は、前記後方ノードを前記要素識別子の種類に関連して左ノードおよび下ノードとして識別し、前記複数の状態は、前記左ノードおよび前記下ノードの状態であることが好ましい。

【００３２】

さらに本発明によれば、文書検索を行うための検索オートマトンを生成するためのコンパイル方法であって、前記コンパイル方法は、

入力された検索式を検索の意味の等価性を保存しつつ逆方向の軸を含む軸およ

び連言や否定を含む論理式を置換することにより状態遷移を生成および登録し、前記後方ノードに対応させて前記後方ノードの複数の状態を記憶装置に記憶させるステップと、

前記後方ノードの複数の状態と遷移条件と少なくとも検索状態と、到達状態のとを対応させて前記記憶装置に登録して検索オートマトンを生成するステップとを含む、コンパイル方法が提供される。

【００３３】

また、本発明によれば、文書検索を行うための検索オートマトンを生成するためのコンパイル方法をコンピュータに実行させるためのプログラムであって、前記プログラムは、コンピュータに対して、

入力された検索式を検索の意味の等価性を保存しつつ逆方向の軸を含む軸および連言や否定を含む論理式を置換することにより状態遷移を生成および登録し、前記後方ノードに対応させて前記後方ノードの複数の状態を記憶装置に記憶させるステップと、

前記後方ノードの複数の状態と遷移条件と少なくとも検索状態と、到達状態のとを対応させて前記記憶装置に登録して検索オートマトンを生成するステップとを実行させる、プログラムが提供される。

【００３４】

さらに、本発明によれば、文書検索を行うための検索オートマトンを生成するためのコンパイル方法をコンピュータに実行させるためのプログラムが記録されたコンピュータ可読な記憶媒体であって、前記プログラムは、コンピュータに対して、

入力された検索式を検索の意味の等価性を保存しつつ逆方向の軸を含む軸および連言や否定を含む論理式を置換することにより状態遷移を生成および登録し、前記後方ノードに対応させて前記後方ノードの複数の状態を記憶装置に記憶させるステップと、

前記後方ノードの複数の状態と遷移条件と少なくとも検索状態と、到達状態のとを対応させて前記記憶媒体に登録して検索オートマトンを生成するステップとを実行させる、記憶媒体が提供される。

【００３５】

また、本発明によれば、要素識別子により要素が区切られた階層構造を有する文書を検索する文書検索システムであって、前記文書検索システムは、

入力される検索式を記憶して構文解析を実行し、前記要素識別子の種類の異なるノードに対して割り当てられる少なくとも２状態を読み込んで状態遷移を可能とする２状態入力オートマトンを生成するコンパイル装置と、

前記２状態入力オートマトンを記憶する記憶装置と、

前記記憶装置から２状態入力オートマトンを読み出して記憶すると共に前記文書を読み込み、前記２状態を識別して状態遷移を可能とするオートマトン評価装置とを含む文書検索システムが提供される。

【００３６】

本発明の前記２状態は、識別された要素識別子に対応して生成された木構造の左ノードおよび下ノードの状態であり、前記２状態入力オートマトンは前記オートマトン評価装置の３つの状態遷移を使用する。

【００３７】

さらに本発明によれば、要素識別子により要素が区切られた階層構造を有する文書を検索するための検索オートマトンを評価する検索オートマトン評価装置であって、該検索オートマトン評価装置は、

コンパイル装置により生成された複数の入力を同時に判断可能とする検索オートマトンを記憶装置から読み込んで記憶する手段と、

前記文書に含まれる前記要素識別子の種類の異なる複数の入力を識別する手段と、

前記識別された入力と前記検索オートマトンに登録された複数の入力とを使用して検索状態を含む３つの状態間で状態遷移を割り当てるための手段と

を含む、検索オートマトン評価装置が提供される。

【００３８】

【発明の実施の形態】

以下本発明を図面に示した実施の形態をもって説明するが、本発明は図面に示した実施の形態に限定されるものではない。また、本発明の理解をより確実に行

うために、以下の実施の形態においてはXML文書を、XPath式を使用して検索を実行するXML文書検索システムを用いて説明を行うが、本発明は、要素識別子を含むいかなる階層化文書に対しても適用することができる。

【００３９】

Ａ：文書検索システムの概略

図１には、本発明の文書検索システムの概略的な構成を示す。本発明の文書検索システム１０は、アプリケーション１２によって使用されるAPIとして機能し、アプリケーション１２からXPath式などにより記述される検索式を読み出して、検索式を検索オートマトンへとコンパイルするためのコンパイ装置１４と、コンパイルされて生成された検索オートマトンを保存しておくための記憶装置１６とを含んで構成されている。さらに、本発明の文書検索システム１０は、検索オートマトン評価装置１８を含んで実装されており、検索オートマトン評価装置１８は、アプリケーション１２が管理する文書と、記憶装置１６に保持された検索オートマトンとを読み出して文書の検索を実行して検索結果を生成し、生成された検索結果を記憶装置２０へと記憶させている。記憶された検索結果は、アプリケーション１２が読み出しを行うか、検索オートマトン評価装置１８が、アプリケーション１２へと出力することにより、アプリケーション１２において検索結果の利用が可能とされている。

【００４０】

本発明の文書検索システムにおいて使用される各装置手段について、XML文書を検索する場合を使用して具体的に説明する。コンパイル装置１４は、アプリケーション１２が指定したXPath式を読み込んで、例えば図示しないパーサを使用して入力XPath式の構文解析を実行し、XPath式に含まれる軸、述語、各述語の論理積、論理和、論理否定をオートマトンとして登録するべき状態遷移へと変換する。検索オートマトンは、変換された各ノードの状態、遷移条件、遷移の種類、到達状態からテーブルとして構成され、生成された検索オートマトンは、ハードディスクといった記憶装置１６へと記録される。

【００４１】

記憶装置１６は、ハードディスク、フラッシュメモリなどの書き換え可能な記

憶媒体を含んで構成されていて、生成された検索オートマトンを保持している。この記憶装置１６に保持された検索オートマトンは、後述する検索オートマトン評価装置へと渡されて、検索オートマトン評価装置による文書の検索が可能とされている。

【００４２】

本発明の検索オートマトン評価装置１８は、状態機械として機能し、検索オートマトンと検索を実行するためのXML文書とを読み込み、構文解析を実行し、XML文書の評価を実行する。検索オートマトン評価装置１８は、記憶装置１６に保持された検索オートマトンを使用して、XML文書を検索し、検索されたXML文書のノードを記憶装置２０へと格納し、アプリケーション１２へと渡している。検索オートマトン評価装置１８は、XPath式で指定されるXML文書においてXpath式で指定されるノードが検索されると検索状態へと遷移する。遷移が生じた時点で評価を行っているノードは、記憶装置２０へと記録され、出力されるまで保持される。

【００４３】

図２は、本発明の文書検索システムにおける状態遷移を示した図である。同時に図２においては、検索状態および終了状態において文書検索システムが実行する処理を示している。図２に示されるように、本発明の文書検索システムは、初期状態I、終了状態F、および検索状態「＊」の３つの状態の間で状態遷移を行う。XPath式が入力され、検索が開始すると、初期状態Ｉは、終了状態Ｆまたは検索状態＊へと遷移し、その状態を変化させる。検索状態＊では、検索状態Ｆへの遷移を生じさせたノードを記憶手段へと記憶させ、本発明の特定の実施の形態では、終了状態Ｆへと遷移するまでの間に検出されたノードを記憶手段に蓄積させる。終了状態Ｆでは、蓄積された検索されたノードをアプリケーションへと渡し、アプリケーションからの検索要求に対する出力処理を実行する。

【００４４】

Ｂ：文書検索システムを構成する装置・ソフトウエア手段

以下、本発明の文書検索システムを構成する図１において説明した各構成装置についてより詳細に説明する。

１．検索オートマトン

本発明においては、検索オートマトンは、アプリケーションにより入力されたXPath式をコンパイルして、各ノードの状態、遷移条件、遷移の種類、到達状態からテーブル構造として構成され、状態遷移を表現するエントリ(以下、状態遷移)の集まりから構成される。状態遷移は、後方ノードに関する複数の状態の組から前方ノードに関する状態への遷移を表現する。通常XML文書のような木構造のノードに対しては、「親」、「子」などの用語を使用してその階層構造を表現するのが通常である。本発明においては木構造をストリーム(ノードの列)としてとらえるので「前方」、「後方」という概念で記述する。本発明において後方ノードとは、XML文書を先頭から見たときに閉じタグで閉じられたノードを意味し、それ以前のノードを前方ノードとして定義する。

【００４５】

図３には、本発明において使用することができるXML文書の具体的な実施の形態を示す。図３に示されるように、通常のXML文書は、開きタグと閉じタグとから構成される。本発明において開きタグとは＜html＞で示されるタグを、閉じタグとは、具体的には＜/html＞のようなタグを意味する。開きタグと閉じタグはその間にある内容を子供とする木構造を作るものとして捉えることができる。

【００４６】

本発明において図３に示すbodyタグが閉じられた場合、titleノード、headノード，div ノード、二つのpノードは、すべてbodyノードの後方ノードとして定義される。図４は、図３に示したXML文書を木構造として示した図である。

【００４７】

本発明の定義によれば、図４では、bodyノードの後方ノードは、（１）bodyノードの弟であるheadノードおよびその後方ノード、（２）bodyノードの下にあるdivとp、および図示しないその後方ノードとなる。divノードは、pノードの後方ノードとして定義される。本発明の定義に基づき、bodyノードにとって直近の後方ノードとは、headノードと、二つ目のpノードとされる。これらのノードを、さらに本発明においては、それぞれbodyノードの左ノードおよび下ノードとして参照する。

【００４８】

図５には、本発明の実施の形態として、本発明により生成される左ノード、下ノードから、現在判断しているノードへの状態遷移の集まりからなる検索オートマトンの実施の形態を示す。図５に示した検索オートマトンは、//pで与えられるXPath式についてコンパイル装置により生成される検索オートマトンを示しているにすぎず、本発明者らは、後述するコンパイル方法により生成することができるいかなる検索オートマトンであっても発明の範囲に含まれるものと考えている。図５に示すように、本発明において生成される検索オートマトンは、到達状態と、遷移の種類と、遷移条件と上述した左ノードおよび下ノードの状態とが対として登録されている。なお、.//pで与えられるXPath式の意味は、「どこかに存在するpノードをすべて検索しろ」という指令である。また、図５に示した検索オートマトンは、ｐが成立しないノード、すなわちタグ名がpではないノードに対して検索状態ではない状態０を割り当てる構成とされている。また、図４および以下に説明する検索オートマトンでは、Iが付された０、１は初期状態であり、Ｆが付された０、１は、終了状態を示し、「＊」が検索状態を表す。

【００４９】

図５で示した本発明において生成される検索オートマトンは、一つの状態遷移が上側の行に対応して生成される。この遷移は、左ノードの状態と、下ノードの状態とが入力として与えられた場合の到達状態として示されている。図５を使用してさらに本発明により生成される検索オートマトンの状態遷移を説明すると、図５のテーブルの上側の行では、左ノードと、下ノードの状態として０，１という状態が指定されている。これは、左ノードが０または１であり、下ノードが０または１という状態を取り得る場合に、その時点で評価を行っているノードへの到達状態が０になりうることを意味している。なお、¬pは、左ノードまたは下ノードまたはその時点で判断しているノードの状態がpではないときに状態遷移を生じさせる演算子である。すなわち、図５に示した上側の行で指定される状態遷移では、左ノードおよび下ノードの両方がpではない場合に状態遷移が生じることを意味する。さらに、図５に示した検索オートマトンは、左ノードおよび下ノードの状態を使用して、その時点で判断しているノードの状態遷移を生じさせ

ることを意味している。

【００５０】

また、図５の検索オートマトンにおける下側の行で特定される状態遷移は、左ノードおよび下ノードがpまたはpでないかの状態を取り得、遷移条件anyは、どのようなノードの名前に対しても遷移が生じることを示す演算子である。したがって、下ノードの状態または左ノードの状態がpを含むか否か、またその時点で評価しているノードがpであるかに応じて検索状態「＊」、初期状態Ｉ、終了状態Ｆという状態が生成される。

【００５１】

図６は、本発明による検索オートマトンの他の実施の形態を示した図である。図６（ａ）に示した検索オートマトンの実施の形態は、//p[not(ancestor::div)]で与えられるXPath式、すなわち従来の表現では「pノードであって上流側にdivノードがないもの」を検索する検索式を受けて、コンパイル装置により生成されるストリーム検索のための検索オートマトンである。図６（ａ）に示した検索オートマトンは、pが成立しない、タグ名がpでないノードに対しては検索状態でない状態０を割り当て、かつdivの子孫ノード(divから見てdescendant軸に対応するノード）にもやはり検索状態ではない状態２を割り当てるような動作を実行するオートマトンである。すなわち、タグ名がｐでないか、またはdivの子孫ノードではない（=divを祖先に持つ)ノードは検索されず、//p[not(ancestor::div)]に対応するノードのみが検索される。本発明において、検索オートマトンがXPath式、//p[not(ancestor::div)]の評価を可能とするのは、本発明が左ノードからの情報と、下ノードからの情報とを同時に状態遷移に使用することにある。すなわち、本発明では、横方向の情報と上下方向の情報とを同時に状態遷移に使用することにより、ノードの兄弟関係に関する軸を検索条件に含めることを可能とする。したがって、従来のXPath式の検索を実行するためのストリーム検索においては評価することができなかったfollowing-sibling、preceding-siblingなどを充分に取り扱うことが可能となる。

【００５２】

検索オートマトンについては、これまでNeven（F. Neven, Design and Analys

is of Query Languages for Structured Documents, PhD Thesis, Limburgs Universitair Centrum, 1999）などにより提案されているものの、Nevenの方法は、検索オートマトンの性質などを調べたものであり、検索オートマトンのストリームベースでの文書検索システムへの適用性およびその具体的構成については何ら示唆するものではなく、本発明の２状態の入力を同時に状態遷移に含ませ、検索状態を定義する構成については何ら示唆するものではない。

【００５３】

さらに、本発明の検索オートマトンを採用することにより、従来技術では不可能だった連言的な、XPath式、例えば//d[.//div and .//p]、を検索式として使用することが可能となる。図６（ｂ）に示した検索オートマトンは、子孫および自分自身がdivでない、または子孫および自分自身がｐではないようなすべてのノードに検索状態ではない０を割り当てるような動作をするオートマトンである。すなわち、子孫にdivを持たないノードおよび子孫のｐを持たないノードは検索されず、//d[.//div and .//p]に対応するノードのみが検索される。いずれの場合でも本発明は、上述した検索オートマトンに対して後方ノードの種類を判断し、その状態を検索オートマトンによる状態遷移に含ませることにより、新たな検索状態「＊」を定義することを可能とする。すなわち、本発明は開きタグと、閉じタグとの間の関係を評価に含ませることで、（１）現在のノードの開きタグの直前の状態、（２）現在のノードの下側ノードの状態の組を状態遷移に使用することが可能となるためである。さらに一般的に言えば、「複数の後方ノードの位置づけを判断し、後方ノードに関連する複数の状態を使用して、現在のノードの状態を計算する」という本発明の本質的な構成に基づくものである。

【００５４】

図７は、上述した本発明の本質的原理を、連言的なXPath式、//d[.//div and .//p]）の評価に適用した場合の検索パスを示すことにより説明した図である。図７に示すように、とあるdノードの下にdivノードとpノードとが両方存在することを検索したい場合について考察する。図７では、divノードとpノードとは、それぞれ別の部分木A、Bに属しているものとする。図７に示した実施の形態において、dノードの下にdivノードとpノードとが両方存在するという情報を得るため

には、木Ａ、木Ｂのどちらかで[.//div]が成立し、かつ他方の木では、[.//p]が成立する、いわゆる連言（and）を調べる必要がある。木Aと木Bとから得た情報を組み合わせてdノードに伝播するということは、XPathのストリーム検索に関する従来技術でも実行されていたが、その場合であってもストリーム検索において、木構造に即して評価を実行するために、[.//div]が存在するかまたは[.//p]が存在する(選言：or)というような２つの条件のどちらかが成立することが独立して伝播されるものでしかない。本発明では、複数の後方ノードの状態のを組み合わせ、その組み合わせが状態遷移のエントリにマッチしていた場合に遷移を生じさせる構成を採用するので、いずれかの後方ノードで成立すべき条件(選言的条件：disjucntive condition)の評価結果もすべての後方ノードで成立すべき条件(連言的条件：conjunctive condition)の評価結果でも、区別せずともに前方ノードに渡すことが可能となる。

【００５５】

２．検索オートマトン評価装置

上述したように、検索オートマトンとは、本発明においては検索オートマトン評価装置の状態を検索状態とする。検索オートマトン評価装置は、入力ストリームを使用して検索オートマトンの評価を実行しながら、入力ストリームを解釈して得られた各ノードの状態に応答して、入力ストリームを解釈した際の検索オートマトン評価装置の内部状態を一時的に記憶する。検索オートマトン評価装置は、記憶された内部状態を左ノードおよび下ノードとに分類して判断を実行・記憶し、遷移先の内部状態が検索状態としてラベルし、XPath式に適合するノードが検索された場合に、当該ノードを記憶装置２０へと記憶させる構成とされている。

【００５６】

図８には、上述した機能を実現する本発明の検索オートマトン評価装置１８の詳細な機能ブロック図を示す。本発明の検索オートマトン評価装置１８は、概ね検索オートマトン保持部２２と、入力ストリームを一時的に格納するためのストリーム格納部２４と、ストリーム格納部２４に格納された入力ストリームを読み出して構文解析を行うパーサ２６と、検索オートマトン保持部２２に保持された

テーブルと、パーサ２６により得られた各タグとを使用して、検索オートマトンの評価を実行する評価部２８とを含んで構成されている。

【００５７】

評価部２８は、入力ストリームから閉じタグか開きタグかの要素識別子を識別して、後方ノードを特定し、特定された後方ノードの状態を検索オートマトンを使用して判断し、その結果を使用して検索状態「＊」を含む状態遷移を生じさせる。評価部２８は、検索状態へと遷移したことを判断した場合には、一時的にその状態および状態遷移を生じさせたノードをメモリ３０へと記憶させ、遅延なく入力ストリームから得られる次のタグの評価を実行する。メモリ３０に出力された検索ノードは、記憶装置２０へと出力され、入力ストリームのすべての評価が終了するまで、検索状態とされたノードを蓄積する構成とされている。また、本発明における他の態様では記憶装置２０を検索オートマトン評価装置１８に設けるのではなく、図１に示したアプリケーション１２に配置することもできる。しかしながら本発明は、アプリケーション１２自体を要旨とするものではないので、上述した他の態様については、詳細な説明を行わない。入力ストリームの評価が終了した時点で、蓄積されたノード集合のデータは、アプリケーション１２へと、アプリケーション１２が入力したXPath式に対する検索結果として渡される。

【００５８】

すなわち、本発明の検索オートマトン評価装置は、１．入力ストリームと、２．検索オートマトンとを入力として受け取る構成とされている。従来技術においても同種の評価装置が提案されているものの、本発明で採用する２入力を使用し、検索状態を含んで構成される検索オートマトンを評価する構成を採用するものではない。

【００５９】

図９には、本発明の検索オートマトン評価装置が実行する検索オートマトン評価方法の概略的なフローチャートを示す。図９に示すように、本発明の検索オートマトン評価装置が実行する方法は、ステップＳ１０において、入力ストリームからタグを読み出し、タグのうちの閉じタグおよび開きタグとを判断する。ステ

ップＳ１２においては、得られたタグデータから前方ノードおよび後方ノードを判断する。ステップＳ１４では、左ノードおよび下ノードをさらに判断し、ステップＳ１６においては、左ノードおよび下ノードを使用して検索オートマトンにより評価を実行し、それぞれ対応する状態遷移を生じさせる。具体的には、その時点で評価を行っている開きタグの直前の状態と、その時点で評価を行っている下ノードの状態を読み出してその時点で評価しているノードの状態遷移を算出する。

【００６０】

ステップＳ１８では、状態遷移が検索状態であれば、その時点で評価を行っているノードが検索されたものとして、メモリにノードを検索状態と共に登録し、評価部２８からそれまでのノードデータを消去する。ステップＳ２０では、検索状態へと遷移したソードが見出される毎にメモリに登録する。ステップＳ２２では、入力ストリームをすべて処理したか否かを、例えば、＜/html＞タグを検出することなどにより判断する。入力ストリームをすべて判断した場合(yes)には、ステップＳ２４へと進んで、検索されたノードを、出力して処理を終了する。この場合、対として登録した検索状態を、ノードの木構造における相対または絶対的な位置情報と共に出力することもできる。入力ストリームをすべて評価していない場合(no)には、ステップＳ１２へと戻って評価を繰り返す。図１０には、本発明の評価方法において検索結果を逐次出力を実行する実施の形態を示す。図１０に示した本発明の他の実施の形態では、メモリに検索状態に遷移したノードを蓄積するのではなく、検索状態に遷移したノードを、ステップＳ４０で逐次出力する構成を採用する。

【００６１】

本発明の方法では、左ノードおよび下ノードといった複数の状態の組を使用して前方ノードにおける遷移を生じさせる。本発明において入力状態として使用することができる複数の状態の組は、特に制限はなく、従来のXPath式の評価方法においては困難であった、たとえば、.//divが成立し、しかも.//pが成立するというような、ふたつの条件の組み合わせを判断することができる。以下、より具体的に本発明の検索オートマトン評価装置の動作について、もっとも単純な、./

/pの検索についての処理ステップを具体的に説明する。以下に説明する実施の形態では、入力ストリームを＜html＞aa
＜/html＞とし、.//pに対応する検索オートマトン（図５）を用いて、ストリーム検索を行う。検索オートマトン評価装置における実行ステップは以下のようになる。＜html＞の左には何のタグもないので、開きタグの直前の状態は初期状態０、１である。の左にも、後方ノードは存在しなかったので、開きタグの直前の状態は初期状態０、１である。一方で、開きタグの下には、テキストaaが存在し、この状態では、テキストノードの状態は、まだ初期状態０、１である。

【００６２】

が検索オートマトン評価装置により検出されると、検索オートマトン評価装置は、それまで評価を実行した開きタグの状態を記憶しているので、pノードの左ノードの状態、すなわち開きタグの直前の状態も、下ノードの状態も０、１となりうる状態であることがわかる。このときには、遷移条件¬pは満たされないが、遷移条件anyは満たされるので、図５の検索オートマトンの２行目の状態遷移を適用して、到達状態１に遷移する。このときの到達状態１は、検索状態「＊」なので、pノードが検索されたことをメモリへと出力する。

【００６３】

さらに、＜/html＞が検索オートマトン評価装置に入力された時点でhtmlノードの左ノードの状態が０、１であること、および下ノードの状態が１であることをメモリから読み出し、htmlノードは、遷移条件¬pと遷移条件anyのどちらも満たし、状態０と１という到達状態に遷移する。一方、このhtmlノードは、到達状態1となっても検索状態に遷移させず、単に終了状態なので、到達したすべての状態で検索されているノードのみが出力される。上述した処理によりpノードのみが出力される。

【００６４】

なお、上述した説明における特定の実施の形態においては、検索オートマトンが「到達したすべての状態で検索されているノードのみを出力する」が、本発明では、後述するように「到達した状態のどれかのノードで検索されているノード

のみを出力する」という検索オートマトンも可能である。

【００６５】

３．検索オートマトン生成のためのコンパイル装置

本発明における検索オートマトンへのコンパイル装置は、入力されたXPath式と等価な検索オートマトン(query automaton)を表現する、遷移状態テーブルを生成し、記憶装置１６に記憶させる機能を有する。図１１には、本発明の検索オートマトンを生成するためのコンパイル装置の詳細な機能ブロック図を示す。図１１に示された検索オートマトン生成のためのコンパイル装置は、パーサ３２と、パーサ３２において生成された構文および各種の論理とを保持するメモリ３４と、メモリ３４に記憶されたデータを読み出して、検索オートマトンに含ませる状態の組み合わせを生成するための検索オートマトン生成部３６とを含んで構成されている。検索オートマトン生成部３６は、生成された検索オートマトンを適切な形式で、記憶装置１６へと出力させている。

【００６６】

本発明の検索オートマトンへのコンパイル装置は、入力されたXPath式に対して、以下に記載する置換を、意味の等価性を保ちつつ実行し、最終的に検索オートマトンの遷移と見なせる形式となった場合に、オートマトンの遷移として記憶手段に記憶させる。

（１）XPathにおける、軸であるchild，descendantで例示されるような順方向の軸の状態遷移への置換。

（２）XPathにおける、軸であるparent，ancestorで例示されるような逆方向の軸の状態遷移への置換。

（３）XPathの兄弟方向(following-sibling，preceding-sibling)の軸の状態遷移への置換。

（４）XPathの述語の状態遷移への置換。

（５）XPathの述語の論理積(and)の状態遷移への置換。

（６）XPathの述語の論理和(or)の状態遷移への置換。

（７）XPathの述語の論理否定(not)の状態遷移への置換。

【００６７】

以下、さらに詳細に本発明のコンパイル装置が実行する処理プロセスを説明する。図１２には、本発明のコンパイル装置が実行するコンパイル方法のフローチャートを示す。本発明のコンパイル方法は、ステップＳ５０においてXPath式またはそれに若干の拡張を加えた検索式(以下、一般的に式として参照する。)を、オートマトンにおける状態として登録する。ステップＳ５２において、式をコンパイル装置が処理しやすい形式、例えばBTLP式へと書き換えを行い、登録する。ステップＳ５４において、書き換えの結果、オートマトンの遷移とみなせる形式が生成されたか否かを判断し、遷移と見なせる形式が生成された場合(yes)には、ステップＳ５６でXPath式における遷移に対応させて記憶させる。ステップＳ５８では、式のすべてを変換したか否かを判断する。ステップＳ５８における判断において、すべての式が判断された場合（ｙｅｓ）には、式のすべての要素がオートマトンの遷移に対応するように翻訳されているので、ステップＳ５８において処理を終了させる。また、すべての式が判断されていない場合(no)には、ステップＳ５２へと戻って、再度書き換えを実行させ、式のすべての要素がオートマトンの遷移に対応して翻訳されるまで反復して変換を実行させる。

【００６８】

本発明においていわゆる「式(expression)」は、計算機の中では木構造を用いて表現することができる。式の書き換えをくりかえし、オートマトンを構築する手法は、たとえば、Gerth, D. Peled, M. Y. Vardi, and P. Wolper、“ Simple on-the-fly automatic verification of linear temporal logic.”、Protocol Specification Testing and Verification, pages 3 -18, 1995に開示されたアルゴリズムなどが知られている。上述した検索オートマトン評価装置および検索オートマトンを生成するためのコンパイル装置は、コンピュータにおいてプログラムとして実装されることにより、それぞれの機能を実現するようにコンピュータを機能させる。以下、擬似コードを使用して、コンピュータに実装された場合の本発明のオートマトン評価装置、およびコンパイル装置の処理を詳細に説明する。

【００６９】

Ｃ：検索オートマトン評価装置、検索オートマトンおよびコンパイル装置の実装

以下、本発明の検索オートマトン評価装置の実装について、擬似コードと、その擬似コードにより実行される処理を用いて説明する。本発明は、XPath式などを含む一般的な「検索式」としてBTLP式を用いて表現し、かつ以下の実施の形態の説明においては、記法・用語について、以下に示した定義を使用するものとする。

【００７０】

値：以下で説明する擬似コードおよび値は、下記の定義に従う。

＜XML/XMLストリーム＞

BTLP式その他BNF構文で定義された「検索式」および文書を意味する。なお、BTLP式については、後述するBNF構文によって定義する。

【００７１】

BTLP式を含め、BNF構文で定義された式は、すべて計算機の中で木構造を用いて表現することができる。ふたつの式が等しい(=)とは、式が構文的に等しいか、または木構造が構造として等しいということを意味する。

【００７２】

＜整数型の値＞

オートマトンの状態は、対応する状態に識別符号、例えば図５に示した１～３などの符号を付し、符号として数字を使用する場合には整数値の集まりとして計算機上に実現される。また、それ以外の式などの値にその構文的な等価性に応じて、ハッシュテーブルを用いて識別番号を与え、これを整数値で代表させて計算機上に記録することもできる。これらは、いかなる組み合わせで登録して使用することができる。

【００７３】

擬似コード:疑似コードにおける演算、命令の定義：

＜集合演算 ∪ ∩ ＼ ＞

それぞれ集合の和、積、差分を意味する。また、本発明における集合は、有限であるものとする。計算機において有限集合を表現し，集合演算を実施する方法は、順序つきリスト、ハッシュテーブル、 ビット集合などさまざまの構成を使用することができることが知られており、本発明においては、これまで知られた

いかなる実装方法でも使用することができる。

【００７４】

述語論理：∀，∃，⇒，∧，∨，¬

本実施例の擬似コードでは、集合の内延的記法の条件記述および ｉｆ文の条件記述のために述語論理を使用する。

【００７５】

全称限量子（unversal quantifier）∀ は、∀x．P(x) のように用いられ、xは束縛変数であり、Pは任意の述語を意味する。コンピュータでこれを評価するためには、ループによってxがとりうる値の範囲をすべて調べ、P(x) がすべてのxで成立するかどうかを調べる処理に対応する。

【００７６】

存在束縛子（existential quantifier）∃ は、∃ x．P(x) のように用いられ、xとPとは、全称限量子と同じ意味を有する。コンピュータでこれを評価するためには、ループによってxがとりうる値の範囲をすべて調べ、P(x) がいずれかのxで成立するかどうかを調べる処理に対応する。また、本発明においては、全称限量子 ∀ および存在限量子 ∃ で束縛される変数のとりうる値の範囲は有限であるものとする。

【００７７】

∧，∨，¬，⇒ は、それぞれ連言、選言、否定、含意を意味する。P∧Qは、PとQとがともに成立するか否かを判断する。P∨Qは、PとQのどちらかが成立するかを判断する。¬Pは、Pが成立しないか否かを判断する。P⇒Qは、Pが成立しないか、またはQが成立するか否かを判断する。

【００７８】

外延的記法 { x1,..., xn} および内包的記法 { f(x1,..., xn) | P(x1,...,xn)}

：

外延的記法は、束縛変数の集合を羅列的に記述するものであり、内包的記法は、まず束縛変数 x1,..., xnのとりうるすべてての値の組を多重ループを用いて判断し、各組に対して P(x1, ..., xn) が成立するかどうか判定する。成立するならば、f(x1, ..., xn) を評価して値を得、得た値を結果集合に加える。という

手順で有限集合の内部表現として記憶させることが可能となる。

【００７９】

組 <>：

組とは、いくつかの要素からなる集まりを意味する。要素の型は異なっていてよく、<1, "string">や<2, "abc"> は、組を意味する。組は、CのstructやJava（登録商標）のクラスといった構造で実現することができる。

【００８０】

直積×：

直積とは、 集合の間の演算を意味し、{1, 2}x{"string", "abc"}={<1, "string">, <1, "abc">, <2, "string">, <2, "abc">} などように両方の集合から読み出した値の組すべてからなる集合をかえす演算である。直積A×Bは、２重のループ構造によってAとBとの各要素を並べ上げ、各要素の組を生成して、登録し、それを結果集合に加えることにより生成することができる。

【００８１】

proc宣言とreturn文：

以下に説明する本発明の実施の形態では、擬似コードはすべて再帰呼び出しを持つ手続きの定義を使用する。proc宣言は、手続きの宣言と引数の定義を行う。return文は、返り値を返す。これらの機構は、既存のCやJava（登録商標）などのプログラミング言語を用いることで実現することができる。

【００８２】

case文：case文は、BNF構文などで定義された式に対するパターンマッチを行う。例えば、ｎを整数値として、

```
a, b, c ::= a + a
          | a - a
          | (a)
          | n
```

で表現される式ａは、0, (0-2)+1など、多数挙げることができる。case文によるパターンマッチとは、式ａを受け取り、

```
case a
```

```
b + c → ....
b - c → ...
  (b) → ...
  n → ...
      otherwise → ...
esac
```

などのように、式の形によって分岐処理を実行する。上述した場合には、aがb + cの形式、例えば(0-2)+1などであれば、b = (0-2), c = 1 というような代入を行い、その後 .... の部分を実行する。aが(b)の形をした(0-2)である場合には、b = 0-2として ... の部分を実行する。マッチするものがない場合には、otherwiseの部分を実行する。case文のような機構は、例えばSML/NJ(http://cmcell-labs.com.cm/cs/what/smlnj/)などに開示のMLといった関数型言語にはすでに存在するが、他の言語を使って、同様の機能を実現することができる。

【００８３】

for 文：

通常のプログラム言語の for 文である。

if 文：

通常のプログラム言語の if 文である。

代入文=:

通常のプログラム言語の代入文である。

【００８４】

不動点計算:本発明において後述する実施の形態における「検索オートマトンへのコンパイル」の方法は、不動点計算アルゴリズムである。不動点計算とは、集合を監視しながらループ計算を行い、集合の要素がこれ以上ふえなくなった時点で計算が終了するアルゴリズムを意味する。集合の内容に変化がなくなったことを知るためには、ひとつの集合のなかに等価な２つの要素が入っていてはならない。このことは値が式である場合は特に重要である。計算機は、式の集合よりも整数値の集合の方が取り扱いやすいので、ハッシュテーブルを用いて式に対して一意に識別番号を付し、式の集合を直接計算機で表現せずに識別番号の集合を

用いて式の集合とみなす手法も用いることができる。

【００８５】

上述した定義を用いて、本発明の検索オートマトン評価装置としてコンピュータにおいて実装できる擬似コードを、図１３に示す。図１３の擬似コードにより示されるように、検索オートマトン評価装置は、再帰呼び出し関数evalを使用して実現することができる。関数evalは、XMLストリームｖを入力として受け取り、処理を行う。また検索オートマトンは、図１３の擬似コードにおいては、＜Q，δ，I，F，Θ＞として与えられ、検索オートマトンは、大域変数として使用される。

【００８６】

１．検索オートマトン評価装置の実装

以下、図９のフローチャートに示した処理に沿って、擬似コードをもって本発明の検索オートマトン評価装置の処理を説明する。

(a)「ストリームからタグを読み込む」および「ストリームをすべて読み終えるまで反復する」処理：これらの処理が擬似コードによってどのように実現されるかについて説明する。以下に説明する検索オートマトン評価装置には、図１４に示されるXML文書が入力され、入力されたXML文書が、パターンマッチによって解析される。検索オートマトン評価装置が図１４に示された入力ストリームを読み込む場合について検討する。この場合パターン、

【００８７】

u ＜σ＞ v ＜/σ＞

に合致する。このパターンにおいて、u、v、σは、変数である。上記のパターンに入力ストリームが合致する場合には、

【００８８】

u = ＜head＞＜title＞Sample＜/title＞＜/head＞

v = A paragraph

＜div＞A paragraph in div

＜/div＞

σ= html

各変数に対応する部分ストリームとして処理を実行する。上記式中、変数uは、html ノードから見たときの左ノードであり、変数vは、下ノードに対応する。

また例えば、

【００８９】

<head><title>Sample</title></head>

を上記のパターンにマッチさせることもできる。 この場合は、

【００９０】

u = ""

v = <title>Sample</title>

σ = head

となる。上記式中、""は、空文字列を示す。空白文字を使用する理由は、headタグの前に要素が存在しないためである。

【００９１】

擬似コードにおける２～６行目では、パターンマッチの繰り返しによって入力ストリームを分解し、読み込んでゆく動作を実行する。具体的には、関数 eval に与えられたストリームvは、３行目のパターンマッチによってuとwとに分解され、その後さらに再帰的にevalによっる評価が実行される。一方で、入力ストリームが文字列s（たとえば""あるいは"Sample"）にまで分解された場合には、擬似コード４行目のパターンにマッチするものとしている。

【００９２】

擬似コードの３行目の評価において、eval(u)を計算する場合には、入力ストリームから、下ノードwの部分は読み込まれていなくてよい。また、eval(w)を計算する場合には、左ノードuの情報を捨ててしまうこともできる。上述した処理により、関数evalは、入力ストリームを一度左から右へと評価して行くだけで、メモリ効率よく処理を終了することができる。

【００９３】

（ｂ）「開きタグあるいは閉じタグの情報を使って状態遷移を行う」処理：この処理が擬似コードによってどのように実現されるかを以下に具体的に説明する。

関数evalは、検索オートマトン＜Q，δ，Ｉ，Ｆ，Θ＞を大域変数として利用する。集合Qは、状態集合であり、集合Iは、初期状態集合であり、集合Fは、終了状態集合であり、集合Θは、検索状態集合であり、δは、遷移関数である。
例えば、図５に示したXPath式、.//p[not(ancestor::div)]に対応する検索オートマトンは、擬似コード上では、下記式の大域変数として表現される。まず各状態集合は下記式で与えられる。

【００９４】

Q = {0, 1, 2, 3}

I = {0, 1, 2}

F = {0, 1, 3}

Θ= {1,3}

上記式中、Qは、すべての状態集合であり、I、F、Θは、それぞれ、I、F、「＊」の種類の状態からなる集合である。遷移関数δは、タグ名σ、左ノードの状態q1、 下ノードの状態q2を引数として受け取り、検索オートマトンのテーブルを検索し、遷移先の状態の集合を返す関数である。関数δは、たとえば

【００９５】

δ(body, 0, 0) = { 0, 1 }

δ(p, 0, 1) = { 1 }

などとして、評価を行い、評価の値を返す。すなわち、bodyタグを遷移条件が許し、左ノードも下ノードも状態０というテーブルのエントリは、検索オートマトンの１行目と２行目にあり、その到達状態の値に対して、それぞれ０と１と返している。また、pタグを遷移条件が許し、左ノードが状態０、右ノードが状態１であるというテーブルのエントリは、 検索オートマトンの２行目にあることを検索し、到達状態１の値を返している。

【００９６】

上述した検索オートマトン評価のステップは、関数evalの３、４行目における状態遷移δ(σ，q’，q”)の計算に対応する。ここで、q’は、左ノードuに対するeval(u)の実行結果を示し、q”は、下ノードwに対するeval(w)の実行結果である。タグ名σは、上述したパターンマッチによって取り出されたその時点で評

価をしているノードのタグ名である。

【００９７】

（ｃ）「遷移した状態が検索状態であれば，その状態が現在のノードを検索したことを記憶しておく」処理および「遷移に際して，それまでに検索されたことが記憶されているノードは遷移先の状態に記憶しておく」処理：これらの処理は、関数evalの６、７、８行目に対応する。以下、上述した両方の処理をまとめて説明する。

【００９８】

疑似コードにあらわれる変数Ξは、関数evalの結果を格納する変数である。関数evalの結果は、各状態に対し、その状態によって選択されたノードを記憶させておくために用いられる。まず、これは６行目のfor文によって、現在の結果変数Ξを読み出し、７行目のif文でそこに検索状態q(∈Θ)が含まれている場合には、現在のノードｖを加えた{＜q，Ｖ ∪ {v}＞}を新たに結果変数Ξに追加登録することにより実現される。たとえば、関数evalが入力の部分ストリームとして

v= A paragraph

を読み込んだとする。このとき、結果変数Ξが＜1，{}＞という状態と、検索ノードの組を含んでいたとする。すなわち、それまで状態１では何のノードも検索されていなかったが、１は検索状態である(１∈Θ)ので７行目を実行することにより新たに＜ 1，{A paragraph

}＞という組が結果変数Ξに加えられる。

【００９９】

次に擬似コードの処理においては説明が前後するものの、以下に「遷移に際して、それまでに検索されたことが記憶されているノードは遷移先の状態に記憶しておく」処理を説明する。擬似コードの３、４行目の動作、すなわちある状態qが与えられたとき、左ノードuおよび下ノードwに対し、各＜q’，x＞ ∈ eval(u) と ＜q”，x＞ ∈ eval(w) に出現するq’、q”で、δ(σ，q’，q”) = q であるような、すべてのq’とq”とについて、下記式の条件を満たすノードxの集合を求めるという操作である。

【０１００】

（条件）∃V. x ∈V ∧ (<q', V> ∈eval(u) ∨ <q'', V> ∈eval(w))

上記式は、qに至るどの遷移元においても検索されるノードxの集合を返す処理を意味する。得られたxの集合と状態qとの組は、結果集合Ξに加えられる。さらに５行目では、vがεの場合には、各初期状態と空集合の組をΞに加える。より具体的に説明すると、入力ストリームとして、

【０１０１】

v = A paragraph

A paragraph in div

を読み込んだ場合、σ= divであり、左ノードu=A paragraph

のまた下ノードw=A paragraph in div

に対する評価結果がそれぞれ、下記式

【０１０２】

eval(u) = {<1, {A paragraph

}>, <2, {}>}

eval(w) = {<1, {A paragraph in div

}, <2, {}>}

であったとする。このとき、δによる遷移には下記式の３つの組みあわせが考えられる。

【０１０３】

δ(div, 1, 1) = {0, 1}

δ(div, 1, 2) = {0}

δ(div, 2, 2) = {2}

ここで、q=1に遷移するためには、図５に示した上側の行の遷移を使うしかないことになる。すなわちq' =1、q" =1の場合しかありえない。このとき、q=1に対して、 ∃V. x ∈V ∧ (<q' , V> ∈eval(u) ∨ <q" , V> ∈eval(w))を満たすxとは、 A paragraph

あるいは A paragraph in div

の双方となる。また、q=2に遷移するためには、q' =2、q" =2の場合しかありえ

ないものの、条件 $\exists V.\ x \in V \wedge (\langle q', V\rangle \in eval(u) \vee \langle q'', V\rangle \in eval(w))$ を満たすxは存在しない。

【０１０４】

一方、q=0に遷移するためには、１行目と２行目の両方を使うことができる。すなわち、 $\forall q', q''.\ \langle q', x\rangle \in eval(u) \vee \langle q'', x\rangle \in eval(w)$ を満たすｘは、A paragraph

のみである。なぜならばq'＝１、q"=2の場合にx=A paragraph in div

は、条件を満たさないためである。 より詳細に言えば、$\langle 2, \{\}\rangle \in eval(w)$ から得られる $V = \{\}$ が、xを含んでいないためである。結果として、

【０１０５】

$\Xi = \{\langle 0, \{$A paragraph

$\}\rangle, \langle 1, \{$A paragraph

, A paragraph in div

$\}\rangle, \langle 2, \{\}\rangle\}$

が得られる。

【０１０６】

本発明では、上述したように、たとえば状態０に至る遷移元すなわち、q'=1、 qq"=1、またはq"=1、q"= 2のどちらを用いても選択されるノードのみを選択する検索オートマトンを用いている。上述したように、関数evalが結果集合Ξに対して、それぞれ状態と検索されたノード集合との組の集合が求められることが示される。なお、「各状態に記憶された検索されたノードを出力する」処理は、下記式の擬似コードにより得られるノード集合を出力することで実行することができる。

【０１０７】

$\{ x \mid \forall q.\ q \in F \wedge \langle q, V\rangle \in eval(v) \Rightarrow x \in V \}$

上記式で示される処理は、終了状態と組とされたどのノード集合にも検索されているxのみを、検索されたとして出力する動作である。

【０１０８】

以下に、本発明の理解を容易にするべく、図６（ａ）で示した検索オートマト

ンを、図１４に示した入力ストリームを使用して評価を行う場合の関数evalの返り値の実施の形態を図示することにより、関数evalの機能を説明する。

(i)まず文字列 "Sample" (あるいは "A paragraph", "A paragraph in div", "" など)を関数evalに入力した場合、eval(ε) = {<0, {} >, <2, {} >} が得られる。図１５には、関数evalの上述した機能により生成される返り値の実施の形態を示す。図１５は、"Sample"を関数evalが読み込んで、状態０、２に遷移したものの、何も検索されなかったことを示す。

【０１０９】

(ii)次に <title>Sample</title> に対する関数evalの処理を図１６に示す。図１６は、状態０、１、２に遷移し、状態１でtitleノードが検索されたことを示している。このため図１６は、<0, {} >, <1, {<title>Sample</title>} >, <2, {} > という3つの組からなる結果を示すものである。図１６の結果が返されるのは、左ノード""からの入力状態０と、下ノード"Sample"からの入力状態０とから、検索状態１への遷移が生じたためである。<head><title>Sample</title></head>に対しては、図１７に示した結果が返される。

【０１１０】

(iii)図１８は、さらに右のbody木に移り、計算を続行した場合のA paragraph、またはA paragraph in div
)に対して生成される結果を示す。図１８に示した結果は、<1, {A paragraph } >, <2, {} > という組からなり、状態０に対しては、¬pという遷移条件のために遷移することができない。しかしながら、A paragraph
A paragraph in div
に対しては、図１９に示されるように、{<0, {A paragraph } >, <1, {A paragraph
, A paragraph in div
} >, <2, {} >} という結果が返される。

【０１１１】

(iv)図２１および図２１には、bodyノードとhtmlノードとについて得られる結果を示す。終了状態は、F = {0, 1} である。このため、最終的には、

$\{ x \mid \forall q.\ q \in F \wedge \langle q, V \rangle \in eval(v) \Rightarrow x \in V \}$

により、０と１の両方で検索される図１９に示したノードで０、１が割り当てられているノードである１つめのpノードが検索されることになる。

【０１１２】

以上説明したアルゴリズムは、 ストリームによってXPathを評価し、ストリームを読み終えた後、検索結果をまとめて出力する。しかしながら、関数evalの評価の途中の段階で、到達状態は {0，1，2} であることと、あるノードが状態１で検索されているが状態０では検索されていない場合であって、現在の状態０から終了状態に到達する遷移が必ず存在するということが判断できる場合には、状態１でしか検索されないノードは、必ず検索されないということの判定が可能である。本発明においては、上述した判断を含ませてストリームを読みながら検索結果を出力することも可能であるし、検索されないノード情報をより早く消去して、ハードウエア資源の効率的利用を達成することが可能となる。

【０１１３】

２．コンパイル装置の実装

本発明のコンパイル装置は、XPath式あるいはそれに若干の拡張を加えた検索式(以下、単に式として参照する。)をオートマトンの状態とみなし、式の書き換えの結果、検索オートマトンの遷移とみなせる形になった場合にそれをオートマトンの遷移に翻訳し、すべてのての様相論理式が書き換えをへてオートマトンの遷移に翻訳されるまで翻訳を繰り返す処理を実行する。以下に説明する本発明のコンパイル装置のコンピュータへの実装の実施の形態においては、式としてXPath式を拡張したBTLP式を用いて説明を行う。本セクションでは、XPathを内部表現であるBTLP式で擬似コードを記述する。BTLP式ϕは、以下のBNF文法で表現できる。

【０１１４】

$b ::= -1 \mid -2$

$m ::= b \mid 1;2 \mid -1;-2$

$\phi ::= a \mid (b) \mid b\phi \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid [m]\phi \mid \langle m \rangle \phi$

BNF文法にのっとった式ϕを計算機の内部で表現するデータ構造は、これまで知

られているいかなる方法を使用することができる。基本の様相b(= 1,2,-1,-2)は、以下のようにXML木を二分木として取り扱うことに対応する。上述した二分木構造を図２２に示す。ここで、１を、「左隣の要素で」、２を、「末子の要素で」、－１を、「末子である場合に親要素で」、－２を、「右隣の要素で」という様相に対応させる。

【０１１５】

一方[m]，＜m＞は、mで指示されるノード集合について「そのどこでも」、または「そのどこかで」を示す様相を意味する。[m]，＜m＞ の説明の前に、それ以外に使用する構文の意味を以下に説明する。

・文字aは、name()="table"や、@titile="xxx"などのXPathの基本的な述語を表すものとする。

・位置判定(b)は、bが示す場所にノードが存在することを示す。あるノードが(1)であるとは、「左隣の要素を持つ」ことを意味し、(2)であるとは、「子要素を持つ」ことを意味し、（－１）とは、「右隣の要素を持つ」ことを意味し、$({-2})$は、「末子でありルートではない」ということを意味する。たとえば図２２（ａ）で、◎の場所では（-1）が成立する。

・bφは、b(= 1，2，-1，-2）が示す場所で、φが成立することを示す。bが示す場所にノードが存在しない場合(つまり、¬(b)が成立する場合)は、bφは必ず成立するものとする。

・φ∧ψは、φとψの双方が成立することを示す。

・φ∨ψは、φとψのどちらかが成立することを示す。

【０１１６】

上述した各演算子の意味づけにおいて、[m]φ,＜m＞φのmについて考察すると、[m]φ,＜m＞φのmには、基本の様相b(= 1，2，-1，-2）が入力できる。まずこの場合を説明する。

・[b]φとはb方向に進み続けるようなパスのどこでもφが成立することを示す。例えば図２２の◎の場所で[-1]aが成立するとは、◎ノードとその兄弟の二つの○ノードとが、すべて基本の述語aを満たすことを意味する。

・＜b＞φは、b方向に進み続けるようなパスのどこかひとつでφが成立すること

を意味する。例えば、上の◎ノードの場所で＜-1＞aが成立するとは、◎ノードとその兄弟の二つの○ノードのうち一つが基本の述語aを満たすことを意味する。また、さらに m = 1;2 または m = -1;-2 の場合も考えられる。これを図２１に示す。図２２（ａ）において1;2は、XPathのdescendant-or-selfに近い様相であるが、子孫だけではなく弟要素もすべて指し示す意味を有する。また、-1;-2は、図２２（ｂ）に示すように、ancestor-or-selfに近い意味を含むが、ルートに至るまでに通る親要素列だけでなく、途中のノードのすべての兄要素も指し示すような様相である。

【０１１７】

より具体的に、XPathの検索オートマトンへの変換を説明するため、P. Gerth, D. Peled, M. Y. Vardi, and P. Wolper, “Simple on-the-fly automatic verification of linear temporal logic.”、In Protocol Specification Testing and Verification, pages 3 -18, 1995に開示の既存アルゴリズムを使用するものとする。例えば、/descendant::pというXPath式を考える。これはBTLP式において、[1;2](p⇒*)と表現することができる。ただし、「＊」は、ノードが選択されることを示すシンボルを意味する。φ⇒ψは、¬φ∨ψの省略標記なので、[1;2](p⇒*)は、[1;2](¬a∨*)と同等である。

【０１１８】

既存アルゴリズムは、「[1;2] φが成立する」ことと「φが成立し、同時に左隣、および末子ノードにおいて[1;2] φが成立すること」とが同値であるという様相論理の性質を利用する点にある。書き下すと下記式で示される等式が成立する。

【０１１９】

$$[1;2](\phi) = \phi \wedge 1[1;2](\phi)) \wedge 2[1;2]\phi$$

この性質を用いて既存アルゴリズムは以下のように動作する。

１．まず式[1;2](¬a∨*) に対応する状態を上の同値関係および(φ∨φ')∧ψとφ∧ψ∨φ'∧ψとが同値であることを使用して、展開する。

【０１２０】

$$[1;2](\neg a \vee *) = (\neg a \wedge 1[1;2](\neg a \vee *) \wedge 2[1;2](\neg a \vee *)) \vee$$

(*∧ 1[1;2](¬a∨*) ∧ 2[1;2](¬a∨*))

得られた∨の両辺に対応する、下記式の２つの状態０、１を生成する。

【０１２１】

¬a ∧ 1[1;2](¬a∨*) ∧ 2[1;2](¬a∨*),

∧ 1[1;2](¬a∨) ∧ 2[1;2](¬a∨*)

【０１２２】

２．次に各状態に対する状態遷移δを生成する。ここで、状態¬a ∧ 1[1;2](¬a∨*) ∧ 2[1;2](¬a∨*)は、１方向(左ノード方向)の状態[1;2](¬a∨*)および２方向の状態（下ノード方向）[1;2](¬a∨*)から到達した状態であると考えることができる。同様に、*∧ 1[1;2](¬a∨*) ∧ 2[1;2](¬a∨*)も状態遷移に変換できる。このようにして生成されたオートマトンを図２３に示す。

【０１２３】

図２３に示したテーブルに４つあらわれる[1;2](¬a∨*)は、展開前の当初の式である。つまりこれをふたたび展開すると再度、

【０１２４】

¬a ∧ 1[1;2](¬a∨*) ∧ 2[1;2](¬a∨*),

∧ 1[1;2](¬a∨) ∧ 2[1;2](¬a∨*)

が得られる。上記式で示されるそれぞれの状態に０，１という符号を割り当てることにより、図２４に示したオートマトンが生成される。

３．初期状態Iを{0，1}、終了状態Fを{0，1}および検索状態Θを{１}で指定する。本発明における終了状態とは、当初の式 [1;2](¬a∨*)から展開された状態０、１を意味する。各状態は様相(b)の条件を含んでいないかぎり初期状態であるため、０、１は、また初期状態ともなる。一方、検索状態は、下記式

【０１２５】

∧ 1[1;2](¬a∨) ∧ 2[1;2](¬a∨*)

で示され、すなわち１である。したがって、図２５に示される検索オートマトンが生成される。

【０１２６】

また、本発明においては、さらに過去の様相である-1、-2を取り扱う変更例を

提供する。より詳細に説明すると、本発明の変更例では、変数qを「状態」と定義し、この状態を、下記式のBNF記法で表現されるデータ構造(リスト構造)とする。

【０１２７】

q ::= ＜Ψ，q＞ | T | ε

ここでΨは、bφ またはσの形をした様相論理式からなる集合である。ここで、q=＜Ψ，q'＞ であるときΨをq.termで、q' をq.base であらわす(特にq1.term = q2.term、かつq1.base=q2.baseならば、q1=q2とする。)。また、q.succ1、q.succ2、q.prec1、q.prec2などは状態ｑに(補助的に)付随する値であるが、これらはq1=q2の判定には関わらない。q.subは、qをbaseとする状態の集合、つまりq.sub=

{q' |q' .base = q}と定義することができる。

【０１２８】

最後に大域的に有効である変数は「状態の集合」Q，P，S，Resolvedおよび「状態の３つ組の集合」AD，AU，BD，BU，CU，CD である。状態の３つ組の集合は、オートマトンにおける左ノード状態、下ノード状態および到達状態の組に相当する。本発明において説明するコンパイル装置のアルゴリズムは、main，expand，extendの３つの関数で定義される。

【０１２９】

図２６には、本発明における検索オートマトンを生成する関数mainが実行する処理のフローチャートを示す。本発明の検索オートマトンを生成する関数mainの処理においては、ステップＳ７０において状態集合を初期化し、ステップＳ７２で、これから計算しなくてはならない３つ組の集合であるCD（S-extendにより計算される）およびCU（P-extendで計算される）が、nullかどうかを判断する。nullである場合(yes)は、ステップＳ８２でAD、AUから検索オートマトンを生成し、ステップＳ８２において処理を終了する。ステップＳ７２の判断において、nullでない場合(no)には、ステップＳ７４においてCDがnullが否かを判断し、nullでない場合(no)には、CDがnullとなるまでステップＳ７８において各CDからAD、BD、CUを計算する。CDがnullとなった場合(yes)には、ステップＳ７６においてC

U=nullか否かを判断し、nullでない場合(no)にはステップＳ８０に進んで、CU=nullとなる（計算するべく状態の３つ組がなくなる）まで各CUからAU、BU、CDを計算し、最終的な検索オートマトンの生成を実行する。

【０１３０】

図２７には、「書き換えの結果,、オートマトンの遷移とみなせる形になったときに、それをオートマトンの遷移に翻訳する。」処理を実行する関数mainの疑似コードを示す。関数mainは、関数S-extendとP-extendとを呼び出し、新しい状態を構築しながら検索オートマトンを構築する。関数mainの返り値は、オートマトンの状態の集合、終了状態と遷移の集合の組を与える。検索オートマトンの検索状態は、$\Theta = \{q \mid * \in q.term\}$として定義される。

最初の関数expandは、「式の書き換えをおこなう」処理を実行する。図２８には、関数expandの疑似コードを示す。

【０１３１】

図２８に示した疑似コードにおいては、$b \subseteq m$は、$1 \subseteq 1;2$ のように定義される。また、$\bigcup\_\{b \subseteq m\}$ ... とは例えば $m = 1;2$である場合には、下記式、

【０１３２】

$expand(\Phi \cup \Psi \cup \{1<1;2>\phi, (2)\}) \cup$

$expand(\Phi \cup \Psi \cup \{2<1;2>\phi, (2)\})$

で記述される集合の和であるとする。

【０１３３】

また、図２８の疑似コードにおいてσは以下のように定義される。

【０１３４】

σ ::= a

| *

| (b)

| $\sigma \wedge \sigma$

| $\sigma \vee \sigma$

| $\neg\sigma$

すなわち様相を含まないBTLP式がσである。この実施の形態のアルゴリズムは、

上述したσから遷移条件を取り出す処理を実行する。

【０１３５】

関数の評価の手法は、用語および擬似コードの解釈の手法にあるとおりである。expand関数は式の集合を受け取り、これを等価な書き換えを通じてbφ、σのどれかの形になるまで変形してゆく。図２８に示した疑似コードに沿って処理を説明すると、３行目のfor文で式をひとつずつ読み出し、５～１２行目のcase文によるパターンマッチで、式をその形に応じて書き換えてゆく動作を実行させる。σの形のものは、６行目でこれ以上書き換えられずに１５行目でreturnされる。また、bφの形のものは １１行目のotherwiseでこれ以上書き換えられずに１６行目でreturnされる。

【０１３６】

図２９には、関数S-extendおよびP-extendの疑似コードを示す。図２９に示された疑似コード(以下では両方をとりまとめてd-extendとして定義している（dは、変数である。)。d-extendは、「XPath式あるいはそれに若干の拡張を加えた式を、オートマトンの状態とみなす」の部分、および「書き換えの結果、オートマトンの遷移とみなせる形になったときにそれをオートマトンの遷移に翻訳する。」)の部分を実行する。

【０１３７】

関数の評価の手法は用語および擬似コードの解釈の手法にあるとおりである．関数が何を行うのかの意味を簡単に説明するとextend関数は、式の集合を受け取り、これから状態の集合を生成する。そしてそれと同時に、検索オートマトンの状態遷移を生成する。具体的な処理を説明すると、１行目の引数(q，Φ)は、それぞれ、ベースとなる状態であり、状態に変換される式の集合である。４行目でexpand関数を呼び出し、Φを等価な書き換えを通じてbφ、σのどれかの形になるまで変形してゆく。６行目のforは、expand関数の結果それぞれを並べあげ、７～１０行目で状態q’を構築する。状態q’がそれまでに生成された状態にないものであれば、１３～３７行目までの処理を実行する。１４～１７行目では、それ以後の計算のために必要なデータを生成する。１８～２４行目は、もし新しい状態q’がS-expandによって作られたものである場合に追加すべき状態遷移を追

加する部分である。図２９に示された疑似コードの２７～３６行目では、もし新しい状態q'が、P-expandによって作られたものである場合に追加すべき状態遷移を追加する部分である。

【０１３８】

本発明において説明した各手段は、中央処理装置（ＣＰＵ）、ＲＡＭ、ＲＯＭなどのメモリ、ハードディスクといった記憶手段などを含むコンピュータまたは情報処理装置において、ソフトウエア的に構成されるソフトウエア・モジュールから構成することができる。また、上述したソフトウエア・モジュールは、本発明において説明した機能を有する限り、図面に示した機能ブロックに対応する構成として含まれるものではなく、異なった機能ブロック構成として構成することができる。さらに、本発明の方法をコンピュータに対して実行させるためのプログラムは、種々のプログラミング言語、例えばアセンブラ言語、C言語、C++言語、Java（登録商標）などを使用して記述することができ、本発明のプログラムを記述したコードは、RAM、ROM、フラッシュ・メモリなどにおけるファームウエアとして含ませることができるし、磁気テープ、フレキシブル・ディスク、ハード・ディスク、コンパクト・ディスク、光磁気ディスク、ディジタル・バーサタイル・ディスク（DVD）といったコンピュータ可読な記録媒体に保持させることができる。

【０１３９】

これまで本発明を図面に記載した具体的な実施の形態をもって説明してきたが、本発明は、上述した特定の実施の形態に制限されるものではなく、種々の変更例および他の実施の形態であっても、本発明の効果を奏する範囲において、これまで知られたいかなる構成要素であっても用いることができる。

【図面の簡単な説明】

【図１】　本発明の文書検索システムの概略的な構成を示した図。

【図２】　本発明の文書検索システムにおける状態遷移を示した図。

【図３】　本発明において使用することができるXML文書の具体的な実施の形態を示した図。

【図４】　図３に示したXML文書を木構造として示した図。

【図５】　本発明の実施の形態として、本発明により生成される左ノード、下ノードから、現在判断しているノードへの状態遷移の集まりからなる検索オートマトンの実施の形態を示した図。

【図６】　本発明による検索オートマトンの他の実施の形態を示した図。

【図７】　本発明の本質的原理を、連言的なXPath式、//d[.//div and .//p])の評価に適用した場合の検索パスを示すことにより説明した図。

【図８】　本発明の検索オートマトン評価装置１８の詳細な機能ブロック図。

【図９】　本発明の検索オートマトン評価装置が実行する検索オートマトン評価方法の概略的なフローチャート。

【図１０】　本発明の評価方法において検索結果を逐次出力を実行する実施の形態を示した図。

【図１１】　本発明の検索オートマトンを生成するためのコンパイル装置の詳細な機能ブロック図。

【図１２】　本発明のコンパイル装置が実行するコンパイル方法のフローチャートを示した図。

【図１３】　本発明の検索オートマトン評価装置としてコンピュータにおいて実装できる擬似コードを示した図。

【図１４】　検索オートマトン評価装置に入力されるXML文書の実施の形態を示した図。

【図１５】　関数evalの上述した機能により生成される返り値の実施の形態を示した図。

【図１６】　関数evalの上述した機能により生成される返り値の実施の形態を示した図。

【図１７】　関数evalの上述した機能により生成される返り値の実施の形態を示した図。

【図１８】　関数evalの上述した機能により生成される返り値の実施の形態を示した図。

【図１９】　関数evalの上述した機能により生成される返り値の実施の形態を示した図。

【図２０】　関数evalの上述した機能により生成される返り値の実施の形態を示した図。

【図２１】　関数evalの上述した機能により生成される返り値の実施の形態を示した図。

【図２２】　基本の様相b(= 1,2,-1,-2)を取り扱う際のXML木を二分木としての取り扱いを示した図。

【図２３】　本発明のコンパイル方法により生成されたオートマトンを示した図。

【図２４】　本発明のコンパイル方法により生成されたさらに処理が進行したオートマトンを示した図。

【図２５】　本発明のコンパイル方法により最終的に生成された検索オートマトンを示した図。

【図２６】　本発明における検索オートマトンを生成する関数mainが実行する処理のフローチャートを示した図。

【図２７】　本発明における関数mainの疑似コードを示した図。

【図２８】　式の書き換えをおこなう処理を実行する関数expandの疑似コードを示した図。

【図２９】　関数S-extendおよびP-extendの疑似コード(まとめてd-extendとして示す。）を示した図。

【符号の説明】

１０…文書検索システム

１２…アプリケーション

１４…コンパイル装置

１６…記憶装置（検索オートマトン）

１８…検索オートマトン評価装置

２０…記憶装置（検索結果）

２２…オートマトン保持部

２４…ストリーム格納部

２６…パーサ

２８…評価部

３０…メモリ

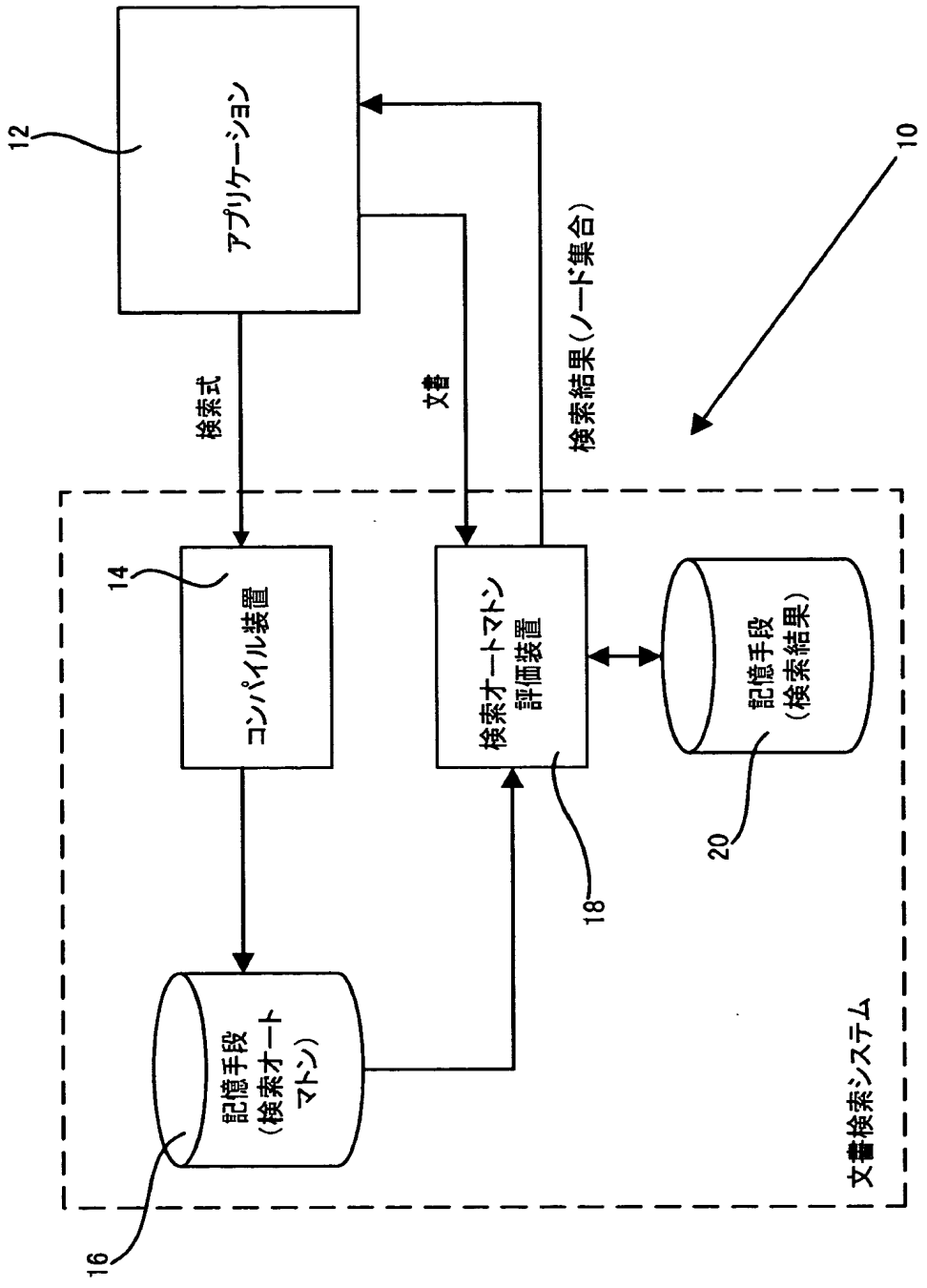
３２…パーサ

３４…メモリ

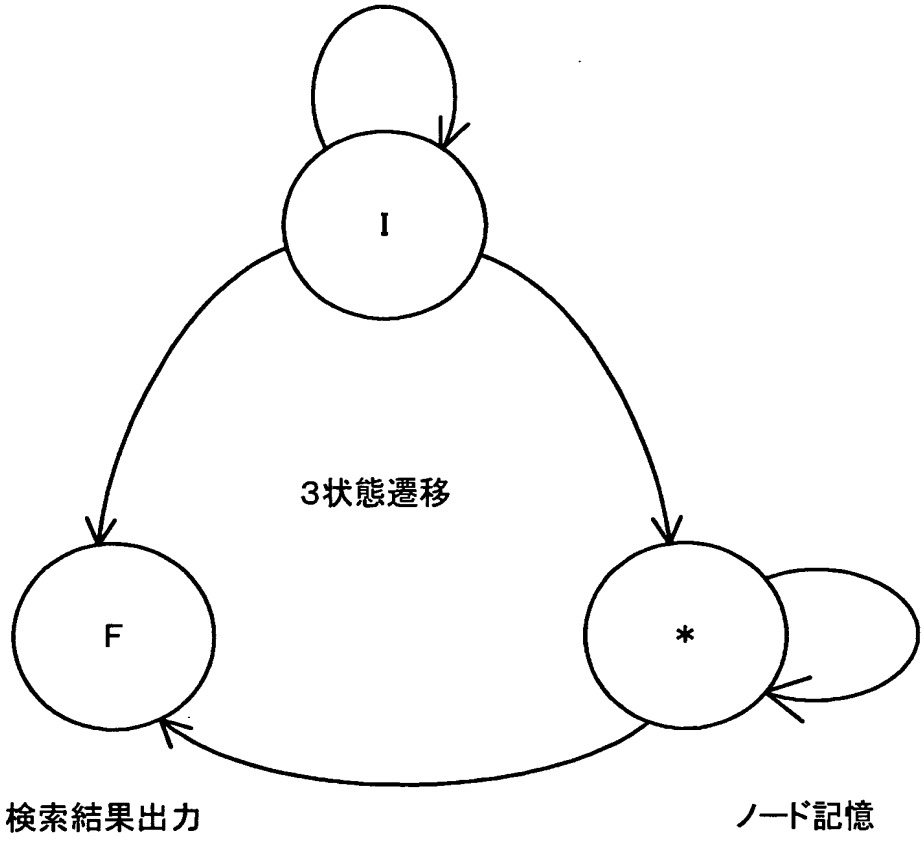
３６…オートマトン生成部

【書類名】　図面

【図１】


12
アプリケーション
10
検索式
文書
検索結果（ノード集合）
14
コンパイル装置
検索オートマトン
評価装置
記憶手段
（検索結果）
20
18
記憶手段
（検索オート
マトン）
16
文書検索システム

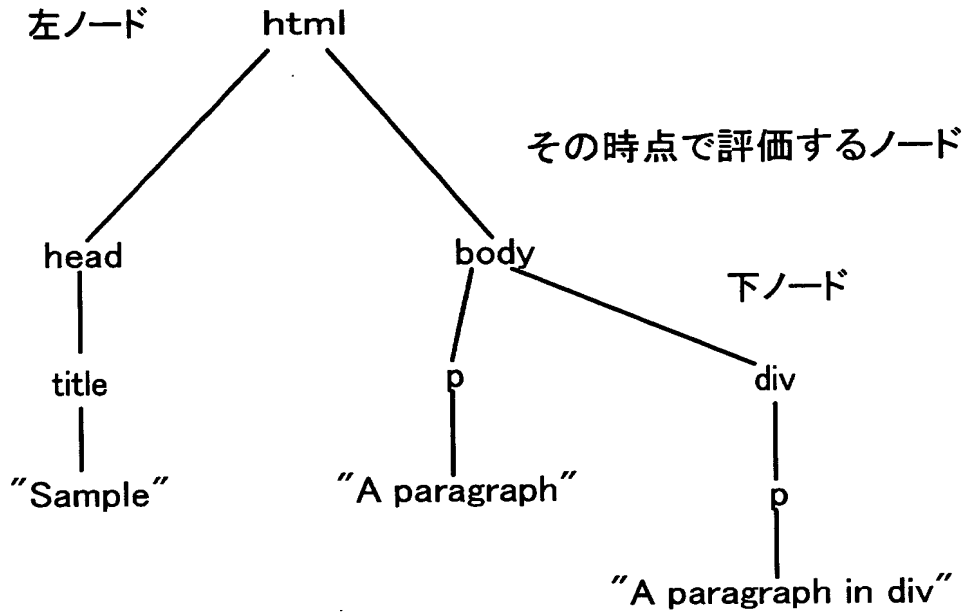
【図２】



【図３】

```
<html>
  <head><title>Sample</title></head>
  <body>
    <div>
      <p>A sample document</p>
    </div>
    <p>A sample paragraph in div</p>
  </body>
</html>
```

【図４】



【図５】

到達状態	種類	遷移条件	左ノードの状態	下ノードの状態
0	I, F	¬p	0, 1	0, 1
1	*, I, F	any	0, 1	0, 1

【図６】

//p[not(ancestor::div]に対する検索オートマトン

到達状態	種類	遷移条件	左ノードの状態	下ノードの状態
0	I, F	¬p	0, 1	0, 1
0	I, F	div	1	2
1	*I, F	any	2	0,1
2	I	any	0,1	2

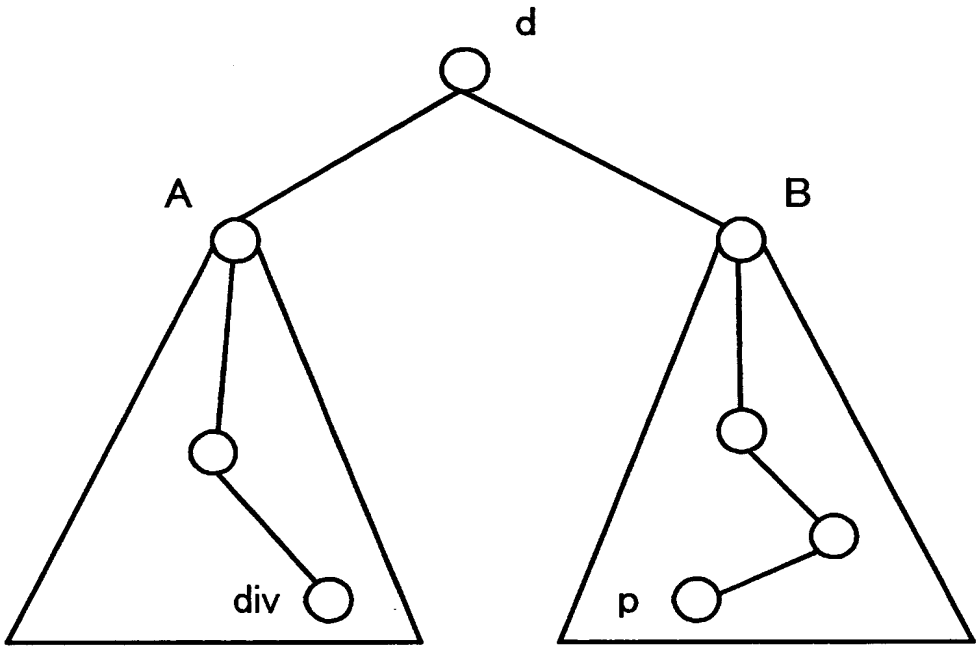
(a)

//*[.//div and .//p]に対する検索オートマトン

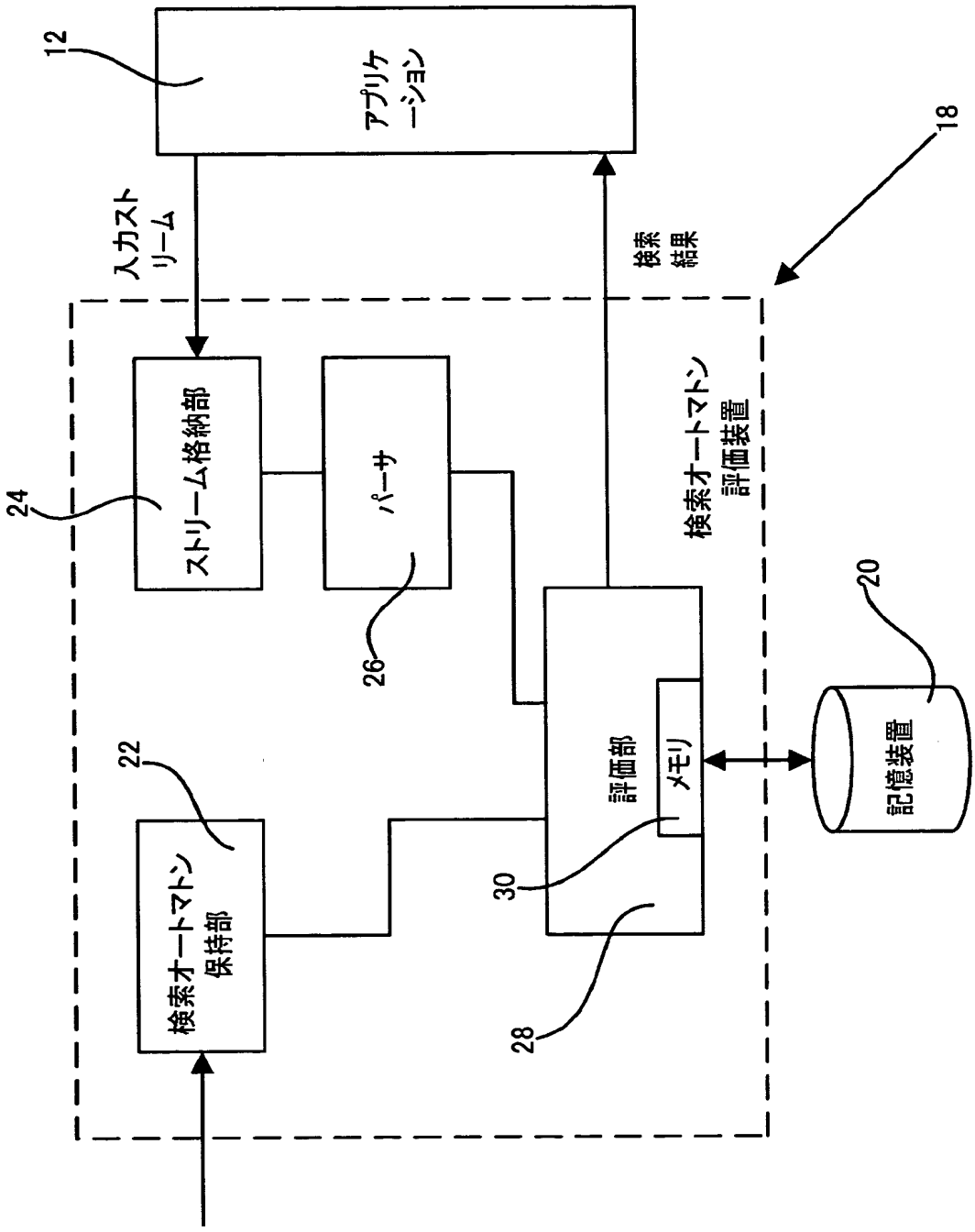
到達状態	種類	遷移条件	左ノードの状態	下ノードの状態
0	F	¬div	0, 1	2
0	F	¬p	0, 1	3
1	*, I, F	any	0, 1	0, 1
2	I	¬div	2	2
3	I	¬p	3	3

(b)

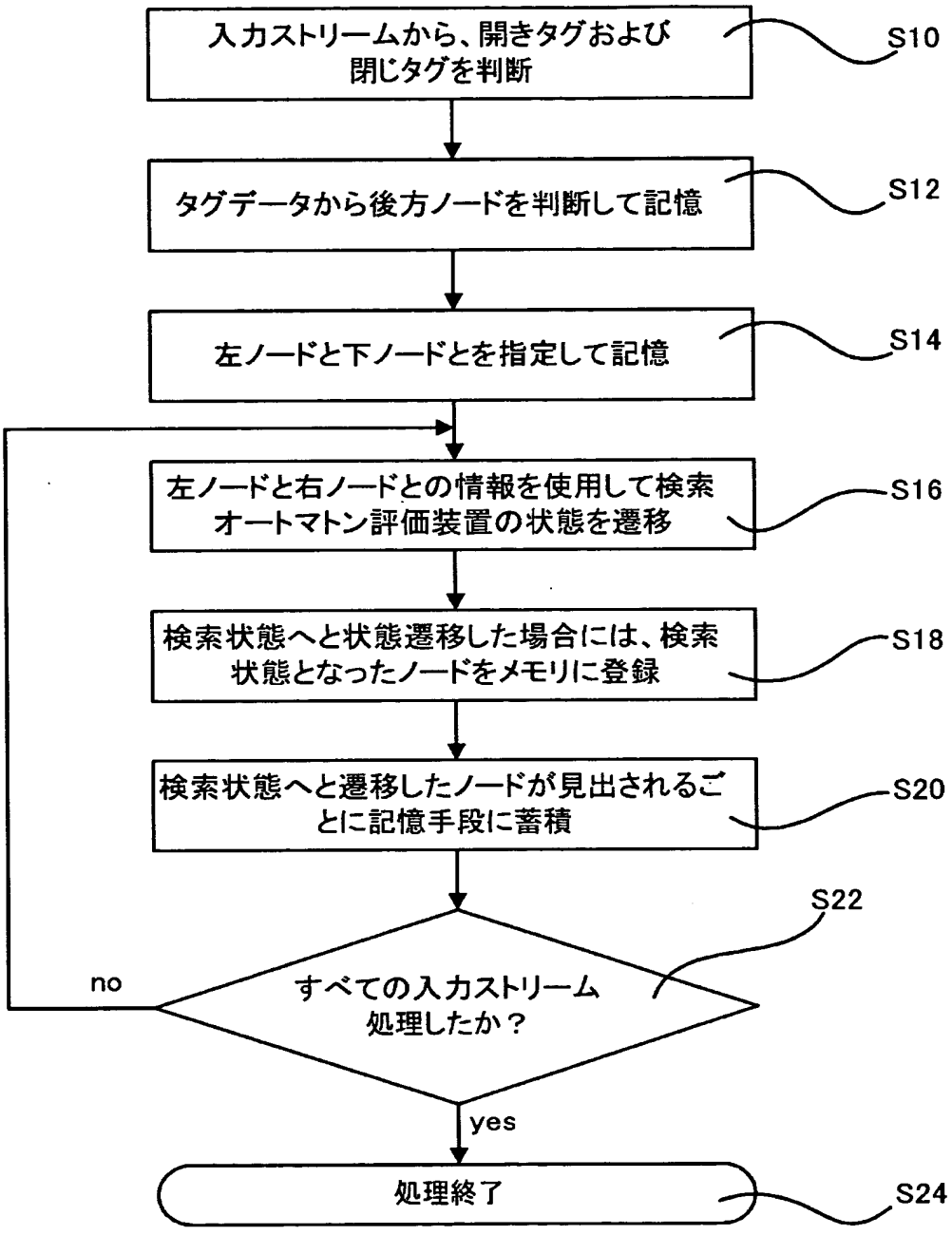
【図７】



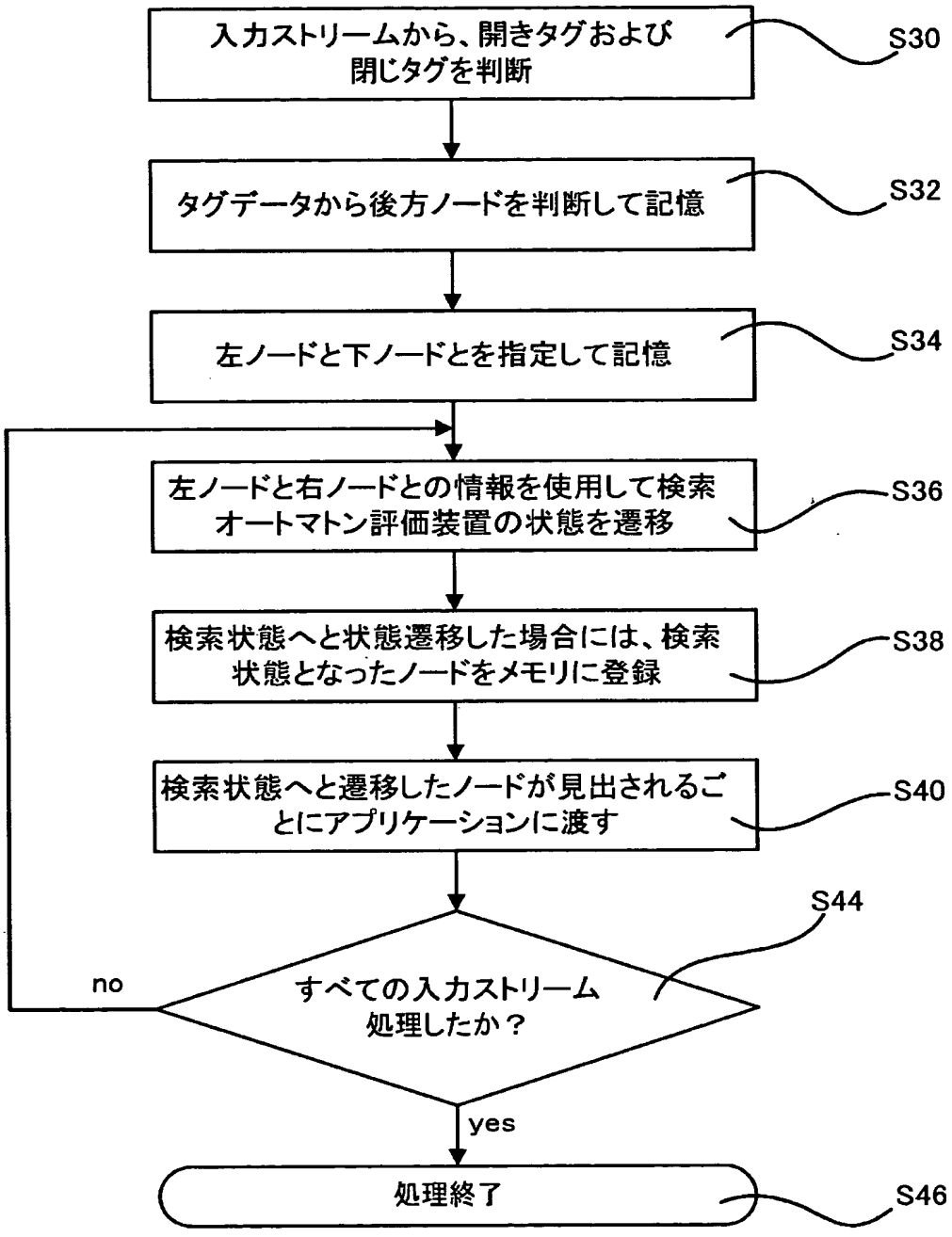
【図８】


12
アプリケーション
入力ストリーム
検索結果
18
検索オートマトン評価装置
ストリーム格納部
24
パーサ
26
評価部
メモリ
20
記憶装置
検索オートマトン保持部
22
30
28

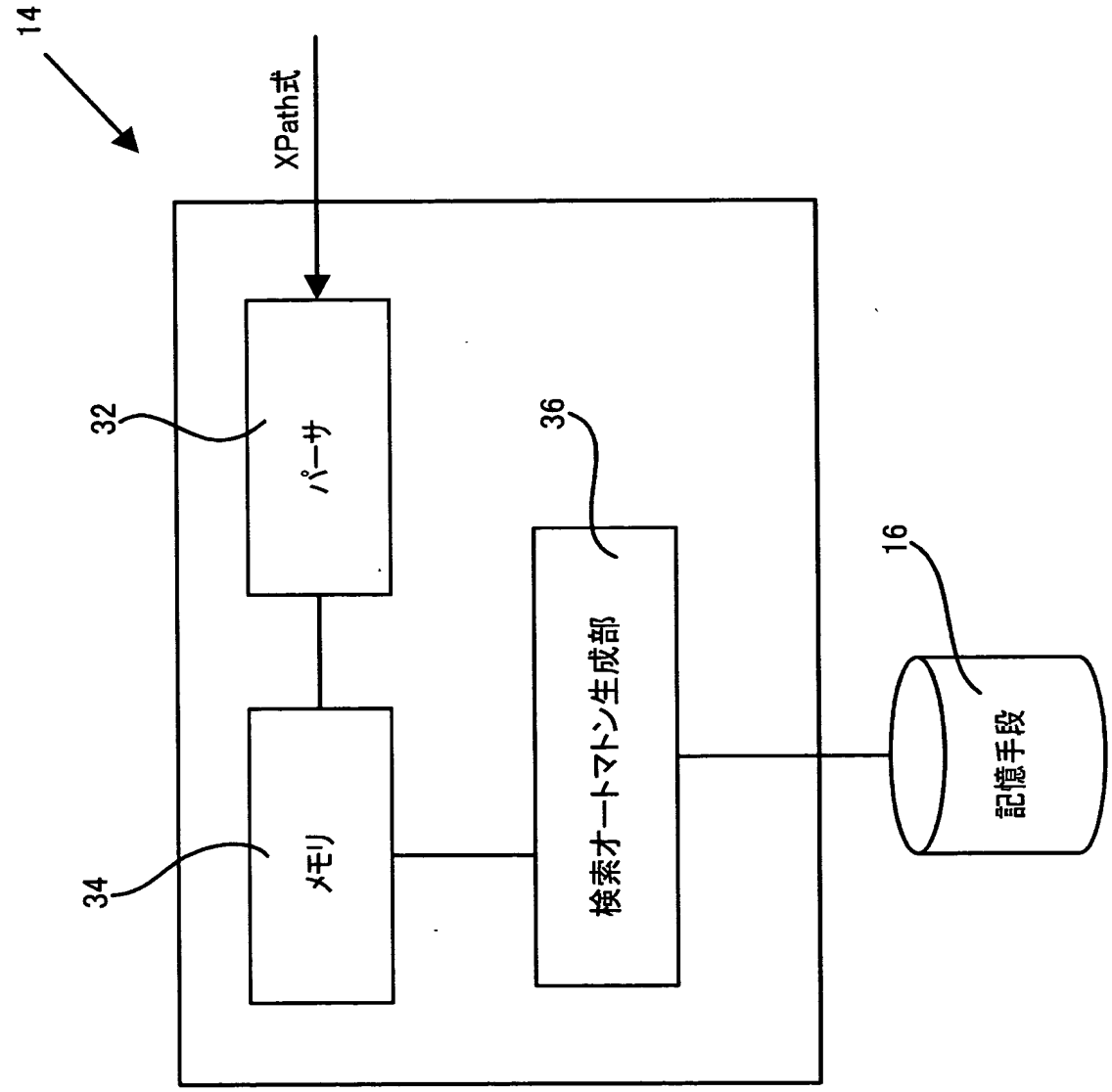
【図９】


入力ストリームから、開きタグおよび閉じタグを判断
S10
タグデータから後方ノードを判断して記憶
S12
左ノードと下ノードとを指定して記憶
S14
左ノードと右ノードとの情報を使用して検索オートマトン評価装置の状態を遷移
S16
検索状態へと状態遷移した場合には、検索状態となったノードをメモリに登録
S18
検索状態へと遷移したノードが見出されるごとに記憶手段に蓄積
S20
S22
すべての入力ストリーム処理したか？
no
yes
処理終了
S24

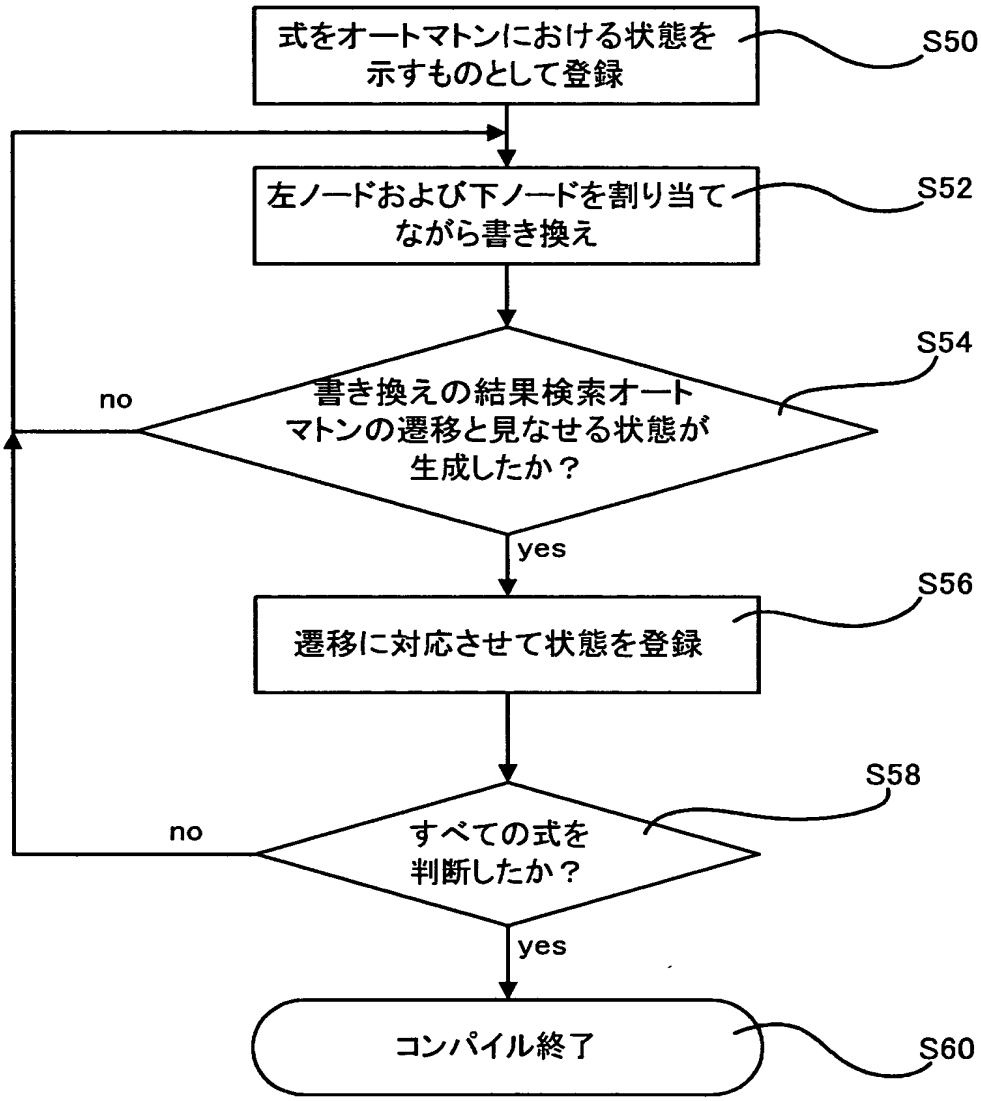
【図１０】


入力ストリームから、開きタグおよび閉じタグを判断
S30
タグデータから後方ノードを判断して記憶
S32
左ノードと下ノードとを指定して記憶
S34
左ノードと右ノードとの情報を使用して検索オートマトン評価装置の状態を遷移
S36
検索状態へと状態遷移した場合には、検索状態となったノードをメモリに登録
S38
検索状態へと遷移したノードが見出されるごとにアプリケーションに渡す
S40
S44
すべての入力ストリーム処理したか？
no
yes
処理終了
S46

【図１１】



【図１２】


式をオートマトンにおける状態を示すものとして登録
S50
左ノードおよび下ノードを割り当てながら書き換え
S52
S54
書き換えの結果検索オートマトンの遷移と見なせる状態が生成したか？
no
yes
S56
遷移に対応させて状態を登録
S58
すべての式を判断したか？
no
yes
コンパイル終了
S60

【図１３】

```
1 proc eval(v)
2   case v
3     u <σ> w </σ> → Ξ := { <q, {x | ∀q', q''. q ∈ δ(σ, q', q'')
4                                   ⇒ ∃V. x ∈V ∧ (<q', V> ∈eval(u) ∨
<q'', V> ∈eval(w))}> | q ∈ Q }
5     s → Ξ := {<q, s> | q ∈ I}
6   esac
7   for <q, x> ∈ Ξ
8     if (q ∈ Θ) Ξ := (Ξ ＼ {<q, V>}) ∪ {<q, V ∪ {v}>} fi
9   rof
10  return Ξ
11 corp
```

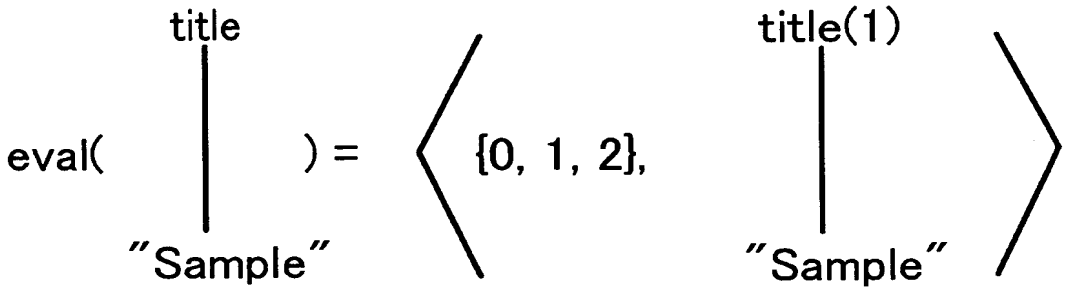
【図１４】

```
<html>
  <head><title>Sample</title></head>
  <body>
    <p>A paragraph</p>
    <div>
        <p>A paragraph in div</p>
    </div>
  </body>
</html>
```

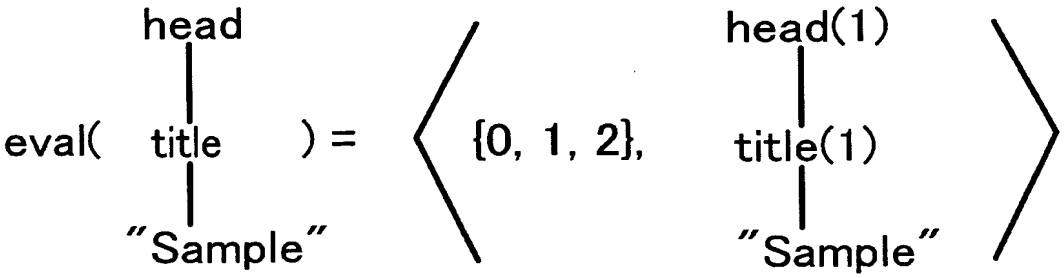
【図１５】

$$\mathrm{eval}(\text{"Sample"}) = \left\langle \quad \{0, 2\}, \text{"Sample"} \quad \right\rangle$$

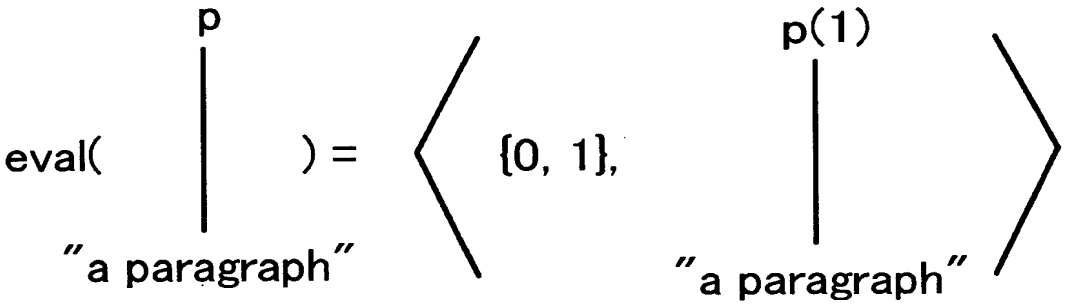
【図１６】



【図１７】

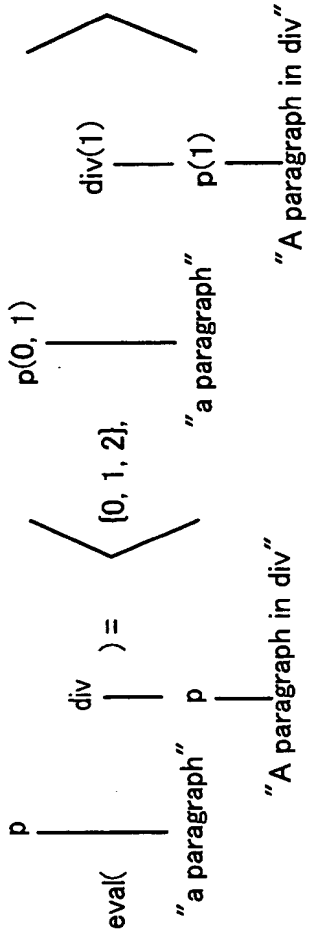


【図１８】

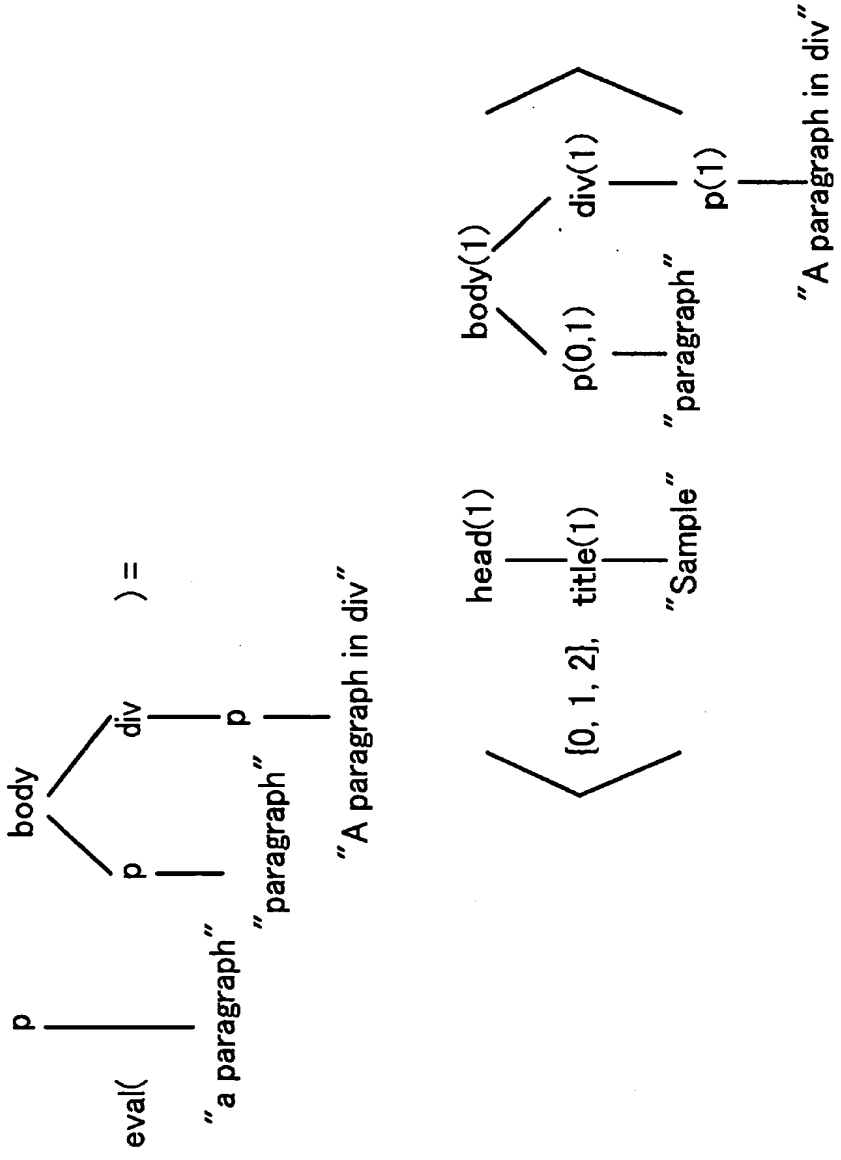


出証特２００３－３００５３６３

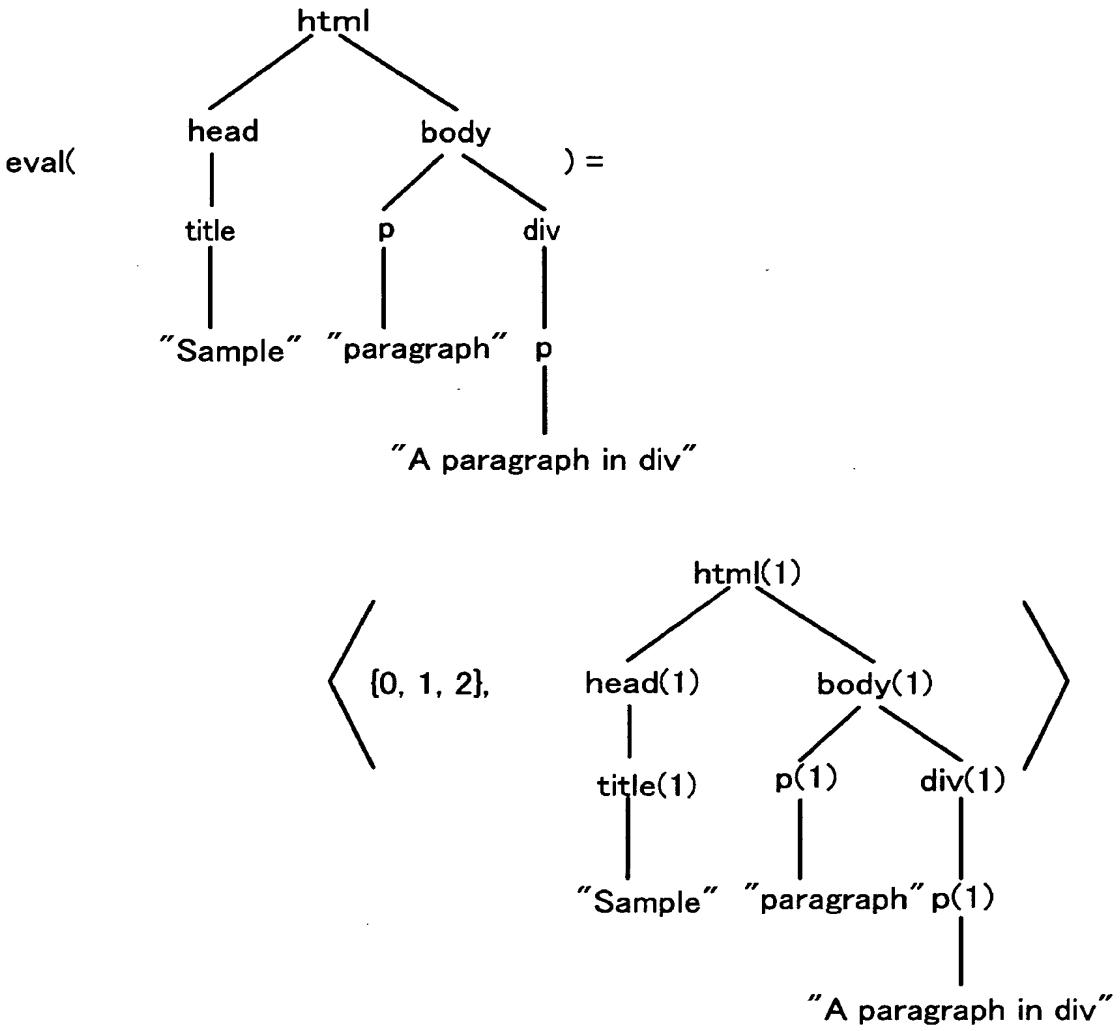
【図１９】


p
eval(
"a paragraph"
div
)=
p
"A paragraph in div"
{0, 1, 2},
p(0, 1)
"a paragraph"
div(1)
p(1)
"A paragraph in div"

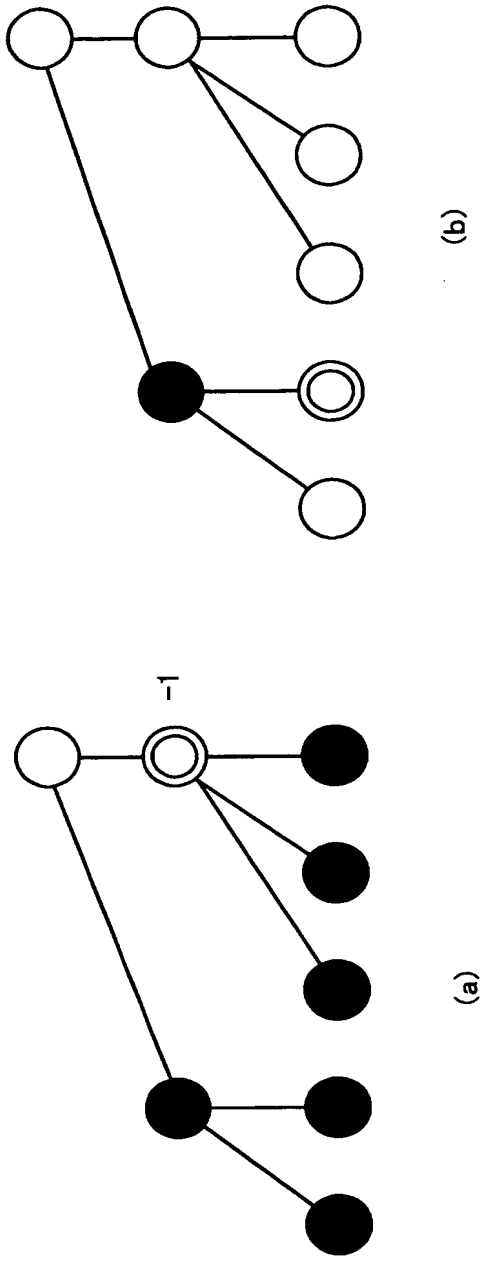
【図２０】


p
eval(
"a paragraph"
)
=
body
p
"paragraph"
div
p
"A paragraph in div"
head(1)
{0, 1, 2},
title(1)
"Sample"
body(1)
p(0,1)
"paragraph"
div(1)
p(1)
"A paragraph in div"

【図２１】


html
head
body
eval(
) =
title
p
div
"Sample"
"paragraph"
p
"A paragraph in div"
html(1)
{0, 1, 2},
head(1)
body(1)
title(1)
p(1)
div(1)
"Sample"
"paragraph"
p(1)
"A paragraph in div"

【図２２】



【図２３】

到達状態	遷移条件	左ノードの状態	下ノードの状態
¬a ∧ 1[1;2](¬a∨*) ∧ 2[1;2](¬a∨*)	¬p	[1;2](¬a∨*)	[1;2](¬a∨*)
∧ 1[1;2](¬a∨) ∧ 2[1;2](¬a∨*)	any	[1;2](¬a∨*)	[1;2](¬a∨*)

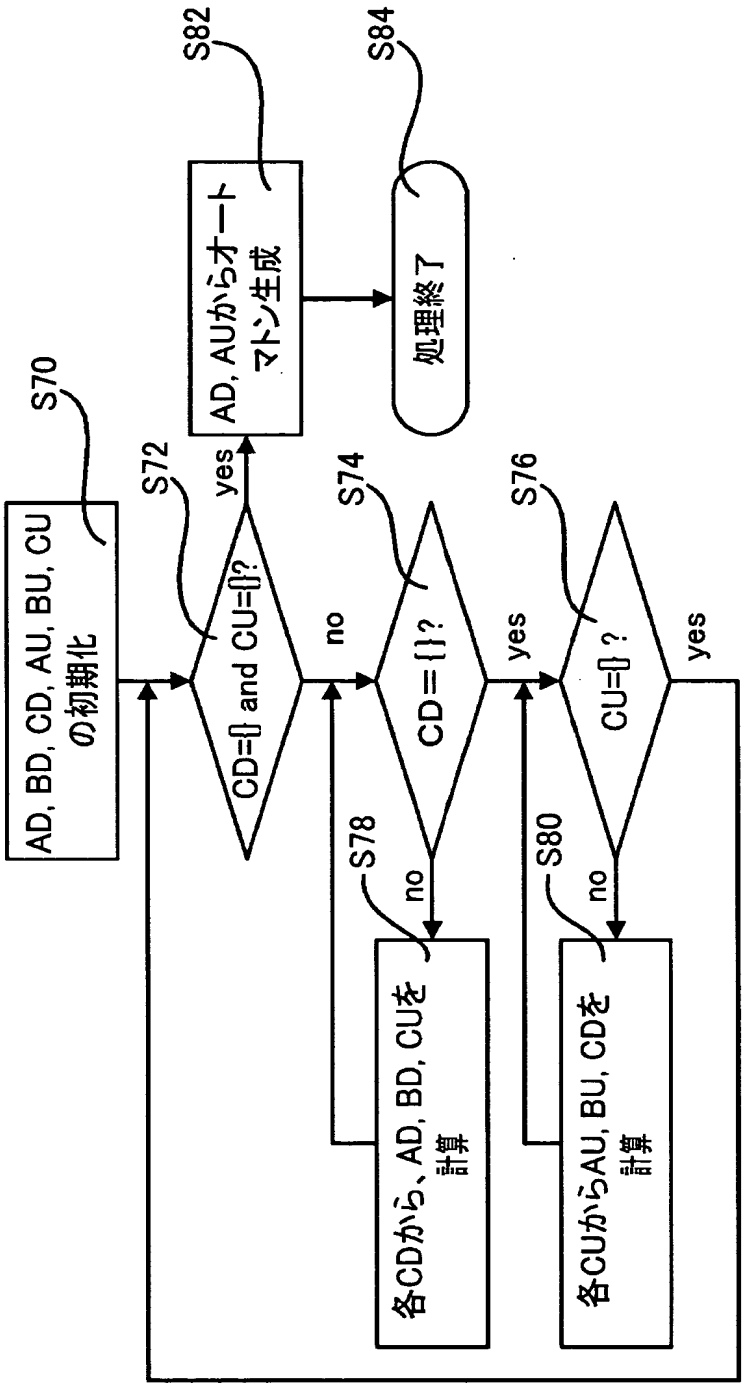
【図２４】

到達状態	遷移条件	左ノードの状態	下ノードの状態
0	¬p	0, 1	0, 1
1	any	0, 1	0, 1

【図２５】

到達状態	種類	遷移条件	左ノードの状態	下ノードの状態
0	I, F	¬p	0, 1	0, 1
1	*, I, F	any	0, 1	0, 1

【図２６】


S70
AD, BD, CD, AU, BU, CU
の初期化
S72
CD={} and CU={}?
yes
no
S74
CD={}?
yes
no
S76
CU={}?
yes
no
S78
各CDから、AD, BD, CUを
計算
S80
各CUからAU, BU, CDを
計算
S82
AD, AUからオート
マトン生成
S84
処理終了

【図２７】

```
1 proc main(φ)
2   Q := P := {T}, S := Resolved := {}
3   AD := BD := BU := CU := CD := {}
4   AU := {<T, T, T>, <T, T, ε>, <T, ε, T>, <T, ε, ε>}
5   T.succ1 := T.succ2 := T.prec1 := T.prec2 := ε.prec1 := ε.prec2 := {}
6   S-extend(T, {φ})
7   while CD ≠ {} or CU ≠ {}
8     for <q0, q1, q2> ∈ CD
9       CD := CD ＼ <q0, q1, q2>
10      X1 := S-extend(q1, q0.succ1)
11      X2 := S-extend(q2, q0.succ2)
12      for <_, q1', q2'> ∈ {q0} × X1 × X2 ＼ AD ＼ BD
13        if ¬∃φ. -1φ ∈ (q1'.term ＼ q1.term) ∧ ¬∃φ. -2φ ∈ (q2'.term ＼ q2.term)
14          then AD := AD ∪ {<q0, q1', q2'>}
15          else BD := BD ∪ {<q0, q1', q2'>}, CU := CU ∪ {<q0, q1', q2'>}
16        fi
17      CU := CU ∪ {q0} × (q1'.sub ＼ Resolved) ×(q2'.sub ＼ Resolved)
18        ∪ {q0} × {q1'} ×(q2'.sub ＼ Resolved) ∪ {q0} × (q1'.sub ＼ Resolved) × {q2'}
19      rof
20    rof
21    for <q0, q1, q2> ∈ CU
22      CU := CU ＼ <q0, q1, q2>
23      X := P-extend(q0, q1.prec1 ∪ q2.prec2)
24      for <q0', _, _> ∈ X × {q1} × {q2} ＼ AU ＼ BU
25        if ¬∃φ. 1φ or 2φ ∈ (q0'.term ＼ q0.term)
26          then AU := AU ∪ <q0', q1, q2>
27          else BU := BU ∪ <q0', q1, q2>, CD = CD ∪ <q0', q1, q2>
28        fi
29        CD := CD ∪ (q0'.sub ＼ Resolved) × {q1} × {q2}
30      rof
31    rof
32  elihw
33   δ := {}
33  for <q0, q1, q2> ∈ AD ∪ AU
34    R0 := {q | q.base = q0 ∧ q ∈ Resolved ∩ S}
35    R1 := {q | q.base = q1 ∧ q ∈ Resolved ∩ P}
36    R2 := {q | q.base = q2 ∧ q ∈ Resolved ∩ P}
37    D := ({q0} ∪ R0) × ({q1} ∪ R1) × ({q2} ∪ R2)
38    for <q0', q1', q2'> ∈ D
39      δ := δ ∪ {(σ1∧...∧σn, q1', q2')→q0'} (σ1, ... ,σn ∈ q0'.term)
40    rof
41  rof
42  F := { q | ∃Φ. Φ ⊆ q.term ∧ Φ ∈ expand(φ) }
43  return <Q, F, δ>
44 corp
```

【図２８】

```
1 proc expand(Φ)
2   Ψ := {}
3  for φ ∈ Φ
4      Φ := Φ ＼ {φ}
5    case φ
6       σ → Ψ := Ψ ∪ {σ}
7       ψ ∧ ψ' → Φ := Φ ∪ {ψ, ψ'}
8       ψ ∨ ψ' → return expand(Φ ∪ Ψ ∪ {ψ}) ∪ expand(Φ ∪ Ψ ∪ {ψ'})
9       [m]ψ → Φ := Φ ∪ {ψ} ∪ {b[m]ψ | b ⊆ m}
10      <m>ψ → return expand(Φ ∪ Ψ ∪ {ψ}) ∪ ∪_{b ⊆ m} expand(Φ ∪ Ψ ∪ {b<m>ψ, (b)})
11      otherwise Ψ := Ψ ∪ {φ}
12     esac
13     fi
14  rof
15  return {Ψ}
16 corp
```

【図２９】

```
1 proc d-extend(q, Φ)
2   if q = ε return {ε}
3   X := {}
4   Ξ := expand(Φ)
5   if ∃Ψ ∈ Ξ. Ψ ⊆ q.term then Ξ := {{}} fi
6   for Ψ ∈ Ξ
7     if ( q ∈ S ∧ d = S) ∨ ( q ∈ P ∧ d = P)
8       then q' := <q.term ∪ Ψ , q.base>
9       else q' := <q.term ∪ Ψ, q>
10    fi
11    X := X ∪ {q'}
12    if ¬(q' ∈ Q) then
13      Q := Q ∪ {q'}
14      q'.succ1 := {ψ | 1 ψ ∈ q'.term ＼ q'.base.term }
15      q'.succ2 := {ψ | 2 ψ ∈ q'.term ＼ q'.base.term }
16      q'.prec1 := {ψ | -1 ψ ∈ q'.term ＼ q'.base.term}
17      q'.prec2 := {ψ | -2 ψ ∈ q'.term ＼ q'.base.term}
18      if d = S then
19        S := S ∪ {q'}
20        if q'.succ1 = q'.succ2 = {} then Resolved := Resolved ∪ {q'}
21         else
22           q'.succ1 := q'.succ1 ∪ q'.base.succ1
23           q'.succ2 := q'.succ2 ∪ q'.base.succ2
24           CD := CD ∪ {<q', q1, q2> | <q'.base, q1, q2> ∈ AU ∪ BU, q1, q2 ∈ Q}
25          fi
26      else
27        P := P ∪ {q'}
28        if q'.prec1 = q'.prec2 = {} then Resolved := Resolved ∪ {q'}
29        else
30          q'.prec1 := q'.prec1 ∪ q'.base.prec1
31          q'.prec2 := q'.prec2 ∪ q'.base.prec2
32          CU := CU ∪ {<q0, q', q''> | <q0, q'.base, q''.base>
                ∈ AD ∪ BD, q0 ∈ Q, q'' ∈ P ＼ Resolved}
33              ∪ {<q0, q'', q'> | <q0, q''.base, q'.base>
                ∈ AD ∪ BD, q0 ∈ Q, q'' ∈ P ＼ Resolved}
34              ∪ {<q0, q', q2> | <q0, q'.base, q2> ∈ AD ∪ BD, q0, q2 ∈ Q}
35              ∪ {<q0, q1, q'> | <q0, q1, q'.base> ∈ AD ∪ BD, q0, q1 ∈ Q}
36        fi
37      fi
38    fi
39  rof
40  return X
41 corp
```

【書類名】　要約書

【要約】

【課題】　文書検索システム、文書検索方法、文書検索を実行するためのプログラム、および該プログラムが記録されたコンピュータ可読な記憶媒体、コンパイル装置、コンパイル方法、コンパイル方法を実行するためのプログラムおよび該プログラムが記録されたコンピュータ可読な記憶媒体、検索オートマトン評価装置を提供する。

【解決手段】　本発明の文書検索は、入力される検索式を記憶して構文解析を実行し、要素識別子の種類の異なるノードから少なくとも２状態を読み込んで状態遷移を可能とする２状態入力オートマトンを生成するコンパイル装置１４と、２状態入力オートマトンを記憶する記憶装置１６と、記憶装置１６から２状態入力オートマトンを読み出して記憶すると共に文書を読み込み、入力された２状態を識別して３つの状態遷移を可能とするオートマトン評価装置１８とを含む。

【選択図】　図１

認定・付加情報

特許出願の番号	特願２００２－２９００５０
受付番号	５０２０１４８４１２６
書類名	特許願
担当官	土井　恵子　　４２６４
作成日	平成１４年１０月　４日

＜認定情報・付加情報＞

【特許出願人】

【識別番号】　390009531

【住所又は居所】　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　ニュー　オーチャード　ロード

【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】　100086243

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内

【氏名又は名称】　坂口　博

【代理人】

【識別番号】　100091568

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内

【氏名又は名称】　市位　嘉宏

【代理人】

【識別番号】　100108501

【住所又は居所】　神奈川県大和市下鶴間１６２３番１４　日本アイ・ビー・エム株式会社　知的所有権

【氏名又は名称】　上野　剛史

【復代理人】　申請人

【識別番号】　100110607

【住所又は居所】　神奈川県大和市中央林間３丁目４番４号　サクライビル４階　間山国際特許事務所

【氏名又は名称】　間山　進也

次頁無

出　願　人　履　歴　情　報

識別番号　　〔３９０００９５３１〕

１．変更年月日　　２００２年　６月　３日
［変更理由］　　住所変更
住　所　　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
氏　名　　インターナショナル・ビジネス・マシーンズ・コーポレーション